



Project S. 5: NGS – Project plan

Ahmed Sayadi

asayadi@univ-catholyon.fr

FIS2 – 2025/2026

Paper summary



RNA-seq and Tn-seq reveal fitness determinants of vancomycin-resistant *Enterococcus faecium* during growth in human serum

Xinglin Zhang, Vincent de Maat, Ana M. Guzmán Prieto, Tomasz K. Prajsnar, Jumamurat R. Bayjanov, Mark de Been, Malbert R. C. Rogers, Marc J. M. Bonten, Stéphane Mesnage, Rob J. L. Willems and Willem van Schaik

Enterococcus faecium is a commensal bacterium in the human gut that is associated with opportunistic bloodstream infections in immunocompromised hospitalized patients. Moreover, it has recently acquired resistance to multiple antibiotics, which represents a big public health concern. However, the growth and survival mechanisms of this opportunistic pathogen in the bloodstream have not been characterized. In this study you will identify what genes allow *E. faecium* to grow in human blood by different profiling techniques based on RNA-Seq and Tn-Seq.

Paper summary



Illumina | PacBio | Nanopore | Genome Assembly | RNA-Seq | Tn-Seq | Differential Expression

The main analyses included in this study are:

- Genome assembly on *E. faecium*
- Differential gene expression of *E. faecium* on human serum against rich medium using RNA-Seq data.
- Identification of genes that contribute to survival and growth in human serum using Tn-Seq data.

What you need to do



Analyses:

1. Genome assembly with PacBio reads.
2. Assembly evaluation.
3. Structural and functional annotation.
4. **Reads preprocessing: trimming + quality check (before and after)**
5. **RNA-Seq reads alignment against assembled genome.**
6. **Differential expression analysis between rich medium and heat-inactivated serum conditions.**

What you need to do



Extra analyses:

Genome assembly with Illumina and Nanopore reads.

- Assembly evaluation (extra methods).
- Plasmid identification.
- SNPs calling.
- Evaluate antibiotic resistance potential.
- Identify essential genes for growth in human serum based on the Tn-Seq data analysis.



- 1. Quality Control (FastQC):** Check raw and trimmed data for quality issues.
- 2. Trimming (Trimmomatic):** Trim low-quality reads and remove adapter sequences.
- 3. Alignment (STAR/BWA):** Align RNA-Seq reads to the reference genome.
- 4. Read Counting (featureCounts/HTSeq):** Count the number of reads mapped to each gene.
- 5. Differential Expression (DESeq2):** Perform differential expression analysis in R.
- 6. Visualization (DESeq2):** Plot results (e.g., MA plot, volcano plot, heatmap).
- 7. Export Results:** Save differential expression results for further analysis.

Data availability



ENA

PRJEB19025: <https://www.ebi.ac.uk/ena/browser/view/PRJEB19025>

SRA

PRJEB19025: https://www.ncbi.nlm.nih.gov/Traces/study/?acc=ERP021008&o=acc_s%3Aa

Read processing – Tools to use



FastQC aims to provide a simple way to check the quality of short reads coming from high throughput sequencing pipelines (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>).

MultiQC compiles quality control metrics from various RNA-Seq processing tools into a single, intuitive report (<https://github.com/MultiQC/MultiQC> <https://seqera.io/multiqc/>).

Trimmomatic aims to trim adapter sequences and low-quality bases from RNA-Seq reads, ensuring clean data for downstream analysis (<http://www.usadellab.org/cms/?page=trimmomatic>).

Read processing – Analysis to do



- Use **FastQC** to check your RNA-seq reads quality (before and after).
 - Report the command line used
 - Explain each parameter used.
 - Run **MultiQC**
 - Interpret the output
- Use **Trimmomatic** to remove low quality reads.
 - Report the command line used
 - Explain each parameter used.
 - Interpret the output

Read processing – Analysis to do



- Use **FastQC** to check your RNA-seq reads quality.

1. Install FastQC

- **Using conda:** `conda install -c bioconda fastqc`
- **Using Homebrew (macOS):** `brew install fastqc`

2. Run FastQC on Single or Multiple Files

- **Single file:** `fastqc sample.fastq`
- **Multiple files:** `fastqc sample1.fastq sample2.fastq`
- **All files in a directory:** `fastqc *.fastq`

3. Check Results

FastQC generates two files for each input file:

- A **.html** report (viewable in any web browser).
- A **.zip** file containing detailed results.

Read processing – Analysis to do



- Run MultiQC.

1. Install MultiQC

- Using conda: `conda install -c bioconda multiqc`
- Using Homebrew (macOS): `brew install brewsci/bio/multiqc`

2. Run FastQC on Single or Multiple Files

```
multiqc .
```

3. Key Sections in the Report:

After running MultiQC, it generates an HTML report and a log file (e.g., MultiQC_Report.html).

- **Summary Statistics:** Overview of metrics, such as read quality, GC content, and adapter contamination
- **Per-Sample Metrics:** Visualizations for each sample, showing read quality, duplication levels, and coverage.
- **Interactive Plots:** interactive plots like sequence quality scores and read length distributions.

Read processing – Analysis to do



- Use **Trimmomatic** to remove low quality reads.

1. Install Trimmomatic

- Using conda: `conda install -c bioconda trimmomatic`
- Using Homebrew (macOS): `brew install trimmomatic`

2. Run Trimmomatic

- Single-End Reads :

```
trimmomatic.jar SE -phred33 input.fastq output_trimmed.fastq SLIDINGWINDOW:4:20 MINLEN:50
```

- Paired-End Reads :

```
trimmomatic.jar PE -phred33 input_R1.fastq input_R2.fastq output_forward_paired.fastq output_forward_unpaired.fastq  
output_reverse_paired.fastq output_reverse_unpaired.fastq SLIDINGWINDOW:4:20 MINLEN:50
```

Read processing – Analysis to do



- Use **Trimmomatic** to remove low quality reads.

3. Common Trimming Steps and Parameters

- **Adapter Removal** (optional):

Provide adapter sequences to trim using the ILLUMINACLIP option.

```
ILLUMINACLIP:adapters.fa:2:30:10
```

- **Sliding Window Trimming**:

Removes bases with average quality below a threshold within a sliding window.

```
SLIDINGWINDOW:4:20
```

- **Minimum Length Filter**:

Discards reads shorter than the specified length.

```
MINLEN:50
```

Read processing – Analysis to do



1. Quality Control with FastQC (Before Trimming)

Run FastQC on Raw Data

```
# Run FastQC on all raw paired-end RNA-Seq data files
fastqc ERR1797969_R1.fastq ERR1797969_R2.fastq \
    ERR1797971_R1.fastq ERR1797971_R2.fastq \
    ERR1797972_R1.fastq ERR1797972_R2.fastq \
    ERR1797973_R1.fastq ERR1797973_R2.fastq \
    ERR1797974_R1.fastq ERR1797974_R2.fastq
```

This will generate **HTML** and **ZIP** files for each paired-end file.

Interpret Results:

Look for quality issues such as:

- **Adapter contamination.**
- **Low per-base quality scores.**

Read processing – Analysis to do



2. Trim Reads Using Trimmomatic

- Install Trimmomatic: `conda install -c bioconda trimmomatic`
- Run Trimmomatic on Paired-End Files

```
# Loop through all paired-end RNA-Seq files and trim them using Trimmomatic
for sample in ERR1797969 ERR1797971 ERR1797972 ERR1797973 ERR1797974; do
    trimmomatic PE ${sample}_R1.fastq ${sample}_R2.fastq \
    ${sample}_R1_trimmed.fastq ${sample}_R1_unpaired.fastq \
    ${sample}_R2_trimmed.fastq ${sample}_R2_unpaired.fastq \
    ILLUMINACLIP:adapter_sequences.fa:2:30:10 SLIDINGWINDOW:4:20 MINLEN:36
done
```

Read processing – Analysis to do



2. Trim Reads Using Trimmomatic

- Install Trimmomatic: `conda install -c bioconda trimmomatic`
- Run Trimmomatic on Paired-End Files

Parameters:

- **ILLUMINACLIP:adapter_sequences.fa:2:30:10**: Trims adapter sequences (ensure you have the correct adapter file).
- **SLIDINGWINDOW:4:20**: Removes low-quality bases using a sliding window (quality < 20).
- **MINLEN:36**: Removes reads shorter than 36 bases after trimming.

Interpret Results:

After trimming, rerun **FastQC** on the trimmed files to ensure improved quality.

Check for:

- Adapter contamination.
- Better per-base quality scores.

Read processing – Analysis to do



2. Trim Reads Using Trimmomatic

- Install Trimmomatic: `conda install -c bioconda trimmomatic`
- Run Trimmomatic on Paired-End Files

Parameters:

- **ILLUMINACLIP:adapter_sequences.fa:2:30:10**: Trims adapter sequences (ensure you have the correct adapter file).
- **SLIDINGWINDOW:4:20**: Removes low-quality bases using a sliding window (quality < 20).
- **MINLEN:36**: Removes reads shorter than 36 bases after trimming.

Interpret Results:

After trimming, rerun **FastQC** on the trimmed files to ensure improved quality.

Check for:

- Adapter contamination.
- Better per-base quality scores.

Read processing - Questions to answer



FastQC

- What is the structure of a FASTQ file?
- How is the quality of the data stored in the FASTQ files?
- How are paired reads identified?
- How is the quality of your data?
- What can generate the issues you observe in your data? Can these cause any problems during subsequent analyses?

Read processing - Questions to answer



Trimmomatic

- How many reads have been discarded after trimming?
- How can this affect your future analyses and results?
- How is the quality of your data after trimming?
- What do the LEADING, TRAILING and SLIDINGWINDOW options do?

Read processing – Questions to answer



Are the majority of samples now within acceptable quality ranges?

Did trimming successfully improve data quality across most samples?

Should any reads be discarded based on length or quality after trimming?

Did trimming improve per-base quality scores?

Has adapter contamination been significantly reduced?

Did you exclude any samples? Explain

Mapping – Tools to use



BWA can align RNA-Seq data, it does not natively support spliced alignments..

STAR (Spliced Transcripts Alignment to a Reference) Primarily designed for aligning RNA-Seq data, which often involves reads that span across exons and introns (<https://github.com/alexdobin/STAR>).

Mapping – Analysis to do



- Map back the reads to the reference genome you assembled previously using **BWA** or **STAR**.
 - Report the command line used
 - Explain each parameter used.
 - Interpret the output

Mapping – Analysis to do



3. Alignment to Reference Genome Using STAR or BWA

STAR is recommended, but **BWA** can be used as an alternative.

Option 1: Using STAR (Recommended for RNA-Seq)

- Install STAR: `conda install -c bioconda star`
- Generate STAR Genome Index (only once per reference genome):
`STAR --runThreadN 4 --runMode genomeGenerate --genomeDir ./STAR_index \
--genomeFastaFiles reference_genome.fasta --sjdbGTFfile reference_annotation.gff`

- Run STAR Alignment on Multiple Paired-End Files:

```
# Loop through all trimmed paired-end RNA-Seq files and align them using STAR  
for sample in ERR1797969 ERR1797971 ERR1797972 ERR1797973 ERR1797974; do  
    STAR --runThreadN 4 --genomeDir ./STAR_index \  
    --readFilesIn ${sample}_R1_trimmed.fastq ${sample}_R2_trimmed.fastq  
    --outFileNamePrefix ${sample}_aligned_ \  
    --outSAMtype BAM SortedByCoordinate
```

done

Mapping – Analysis to do



3. Alignment to Reference Genome Using STAR or BWA

STAR is recommended, but **BWA** can be used as an alternative.

Option 1: Using STAR (Recommended for RNA-Seq)

Parameters:--runThreadN 4: Use 4 threads for parallel processing.

--genomeDir ./STAR_index: Directory to store the STAR index.

--genomeFastaFiles reference_genome.fasta: Path to the reference genome.

--sjdbGTFfile reference_annotation.gff: Path to your **GFF** file from **Prokka**.

--readFilesIn: The paired-end trimmed FASTQ files.

--outFileNamePrefix: Prefix for output files.

--outSAMtype BAM SortedByCoordinate: Output alignment as sorted BAM files.

Mapping – Analysis to do



3. Alignment to Reference Genome Using STAR or BWA

STAR is recommended, but **BWA** can be used as an alternative.

Option 2: Using BWA (Alternative)

- Install BWA: `conda install -c bioconda bwa`
- Create BWA Index for Reference Genome: `bwa index reference_genome.fasta`
- Run BWA MEM for Alignment:

```
# Loop through all trimmed paired-end RNA-Seq files and align them using BWA
for sample in ERR1797969 ERR1797971 ERR1797972 ERR1797973 ERR1797974; do
    bwa mem -t 4 reference_genome.fasta ${sample}_R1_trimmed.fastq
    ${sample}_R2_trimmed.fastq > ${sample}_aligned.sam
done
```

Mapping – Analysis to do



3. Alignment to Reference Genome Using STAR or BWA

STAR is recommended, but **BWA** can be used as an alternative.

Option 2: Using BWA (Alternative)

- Install BWA: `conda install -c bioconda bwa`
- Create BWA Index for Reference Genome: `bwa index reference_genome.fasta`
- Run BWA MEM for Alignment:
Parameters:
 - **bwa mem**: Alignment algorithm.
 - **-t 4**: Use 4 threads.
 - **reference_genome.fasta**: Path to the reference genome.
 - **\${sample}_R1_trimmed.fastq \${sample}_R2_trimmed.fastq**: Paired-end trimmed files.

Mapping – Questions to answer



- What percentage of your reads map back to your contigs? Why do you think that is?
- What potential issues can cause mRNA reads not to map properly to genes in the chromosome? Do you expect this to differ between prokaryotic and eukaryotic projects?
- What percentage of reads map to genes?
- How many reads do not map to genes? What does that mean? How does that relate to the type of sequencing data you are mapping?
- What do you interpret from your read coverage differences across the genome?
- Do you see big differences between replicates?

Post Mapping – Analysis to do



SAMtools provides various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format.

(<https://www.htslib.org/>)

(<https://github.com/samtools/samtools>)

Post Mapping – Analysis to do



4. Convert SAM to BAM and Sort (using SAMtools):

Install SAMtools

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

Convert SAM to BAM

```
samtools view -bS sample1_aligned.sam > sample1_aligned.bam
```

Sort the BAM file

```
samtools sort sample1_aligned.bam -o sample1_aligned_sorted.bam
```

Index the BAM file (needed for counting)

```
samtools index sample1_aligned_sorted.bam
```

Read counting – Tools to use



When performing **differential gene expression (DGE)** analysis after RNA-Seq alignment (e.g., using STAR), the next step is to count the number of reads mapping to each gene. **HTSeq** is one of the most commonly used tools for this task, but there are alternative tools such as **FeatureCounts**.

Read counting – Tools to use



HTSeq is a Python-based tool designed to count reads mapped to genes from **aligned BAM** files (typically generated by RNA-Seq aligners like STAR, HISAT2, or Tophat) (<https://htseq.readthedocs.io/en/latest/>).

FeatureCounts is another excellent tool for counting reads, particularly known for its **speed and memory efficiency**. It's part of the **Subread package** and can process large datasets much faster than HTSeq (<https://subread.sourceforge.net/featureCounts.html>).

Read counting – Analysis to do



- use **HTSeq** to count the number of reads mapping to each gene.
 - Report the command line used
 - Explain each parameter used.
 - Interpret the output

Read counting – Analysis to do



5. Read Counting with HTSeq

- Install HTSeq: `conda install -c conda-forge htseq`
- Run HTSeq to Count Reads for Each Gene:

```
# Loop through all aligned BAM files and count reads per gene using HTSeq
for sample in ERR1797969 ERR1797971 ERR1797972 ERR1797973 ERR1797974; do
    htseq-count -f bam -r pos -s no -t exon -i ID ${sample}_aligned_Aligned.sortedByCoord.out.bam
    reference_annotation.gff > ${sample}_counts.txt
```

Done

Parameters:

- f bam:** Input file format (BAM).
- r pos:** Sort by position.
- s no:** No strand-specific alignment.
- t exon:** Count reads mapping to exons.
- i ID:** Use the **ID** attribute from the **GFF** file as the feature for counting.
- reference_annotation.gff:** The GFF file generated by **Prokka**.



Expression analyses

To compare gene expression between samples, first, we need to count the amount of reads that maps against each gene, and then we determine the differences between them. Once you have the count table with the number of reads mapped to each gene (see “Read counting” section), you can run the differential expression analysis on it:

[**DESeq2**](#) is an R package included in Bioconductor to estimate variance-mean dependence in the counted reads against gene features under different conditions, and test for differential expression based.

DGE – Analysis to do



Install DESeq2 in R:

```
# Install DESeq2 (if not already installed) : install.packages("BiocManager") BiocManager::install("DESeq2")
```

Load DESeq2 and Prepare Data:

```
library(DESeq2)
```

```
# Load the count data (rows are genes, columns are samples)
```

```
count_data<- read.table("sample1_counts.txt", header=FALSE, row.names=1)
```

```
# Prepare sample information (conditions or treatments)
```

```
sample_info<- data.frame( row.names= colnames(count_data), condition = c("control", "treatment", "control", "treatment"))
```

```
# Modify according to your design )
```

```
# Create DESeqDataSet object
```

```
dds <- DESeqDataSetFromMatrix(countData = count_data, colData = sample_info, design = ~ condition)
```

```
# Filter lowly expressed genes (optional but recommended)
```

```
dds <- dds[rowSums(counts(dds)) > 10, ]
```

```
# Run DESeq2 analysis
```

```
dds <- DESeq(dds)
```

```
# Get results
```

```
res <- results(dds)
```

```
# View results
```

```
head(res)
```



Expression analyses

- If your expression results differ from those in the published article, why could it be?
- How do the different samples and replicates cluster together?
- What effect and implications has the p-value selection in the expression results?
- What is the q-value and how does it differ from the p-value? Which one should you use to determine if the result is statistically significant?
- Do you need a normalization step? What would you normalize against? Does DESeq do it?
- What would you do to increase the statistical power of your expression analysis?
- In the metagenomics project, the data doesn't offer enough statistical power for a differential expression analysis. Why not? What can you still tell from the data?

Visualization – Tools to use



Genome visualisation

- [**IGV**](#) is a visualization tool for interactive exploration of large, integrated genomic datasets useful to analyse different data types (short and long reads and genomic annotation).
- **Artemis Comparison Tool (ACT)** is another visualization tool especially designed for displaying pairwise comparisons between two or more DNA sequences (obtained with BLAST).

Visualization – Analysis to do



MA Plot:

```
# MA plot to visualize the distribution of log fold changes vs. mean expression
plotMA(res, main="DESeq2 MA plot")
```

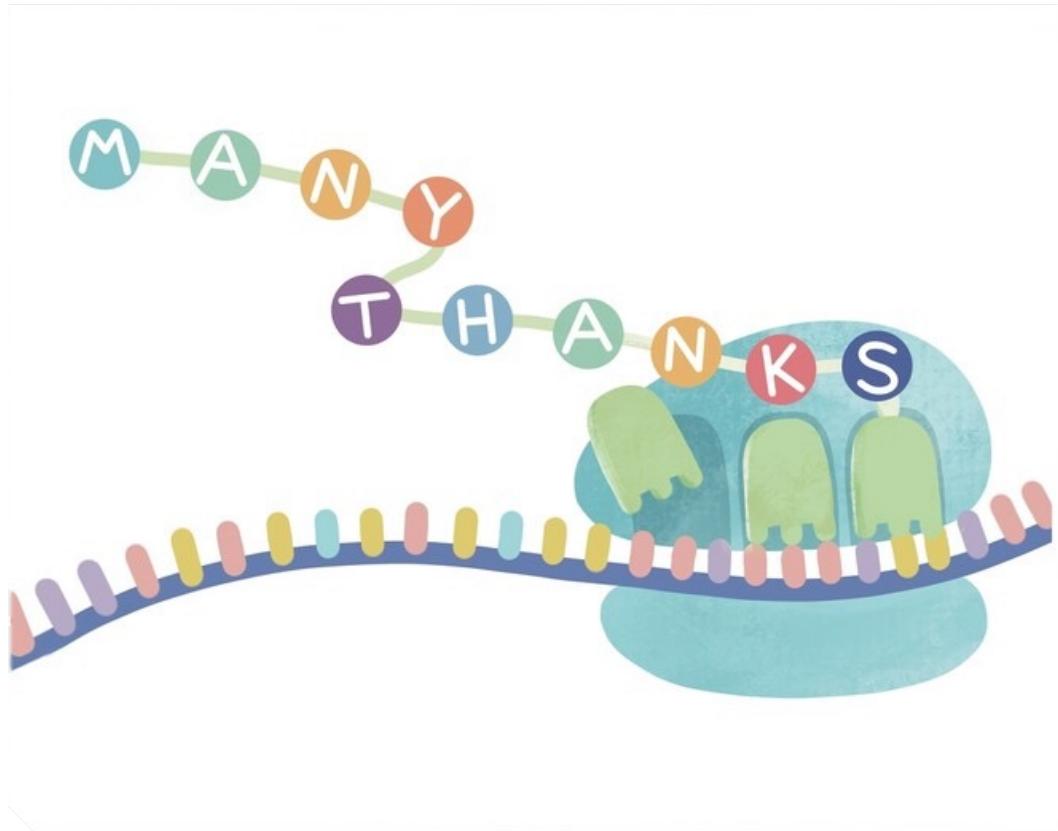
Volcano Plot:

```
# Volcano plot for visualizing significance vs. fold change
library(ggplot2)
ggplot(as.data.frame(res), aes(x=log2FoldChange, y=-log10(pvalue))) + geom_point(aes(color=padj < 0.05)) +
  theme_minimal() + labs(title="Volcano Plot")
```

Heatmap of Gene Expression:

```
# Normalized counts for heatmap
vsd <- vst(dds, blind=FALSE)
select <- head(order(rowMeans(counts(dds, normalized=TRUE))), decreasing=TRUE), 30)
pheatmap(assay(vsd)[select, ], cluster_rows=TRUE, cluster_cols=TRUE)
```

Exporting Results: `write.csv(as.data.frame(res), file="differential_expression_results.csv")`



© TrailMixArt

