Reprohackathon-RNASeq

Tom Bellivier, Tom Gortana, Marie Meier, Donatien Wallaert

November 2025

1 Sommaire

2 Introduction

Reproducibility is a core principle of the scientific method, as it allows independent researchers to validate findings and extend previous work. In bioinformatics and computational biology, achieving reproducible analyses is particularly challenging due to complex software stacks, evolving reference data, and often incomplete documentation. Addressing these issues requires integrated frameworks that combine version control, containerization, workflow management, and clear reporting of analytical steps.

This reprohackathon project focuses on reproducing the RNA-sequencing analysis from Peyrusson et al. (Nature Communications, 2020), which examined intracellular Staphylococcus aureus persisters during antibiotic exposure. Persister cells are phenotypic variants that transiently tolerate antibiotics without carrying genetic resistance mutations, making them important contributors to treatment failure and recurrent infections.

2.1 Biological Context: *Staphylococcus aureus* Persisters and Antibiotic Tolerance

Staphylococcus aureus is a major human pathogen causing infections that range from mild skin disease to severe invasive syndromes. Intracellular survival of *S. aureus* within host cells, such as macrophages, is believed to promote infection recurrence by protecting bacteria from immune responses and antibiotic activity. Under intracellular stresses, a subpopulation can adopt a persister state characterized by growth arrest and high tolerance to bactericidal antibiotics.

Peyrusson et al. used single-cell fluorescence-based approaches to show that intracellular *S. aureus* surviving antibiotic treatment display biphasic killing kinetics, with most bacteria rapidly dying and a minority persisting for prolonged periods. The persister phenotype was stable under antibiotic pressure but reversible once the drug was removed, as non-dividing cells resumed growth when the stress was lifted.

Transcriptomic profiling in the same study revealed that intracellular persisters remained metabolically active but exhibited a markedly reprogrammed gene expression pattern. RNA-seq identified 1,477 differentially expressed genes between intracellular persisters and control bacteria, including induction of stringent response, cell wall stress, SOS DNA damage, and heat shock pathways, which together were associated with multidrug tolerance and cross-tolerance to unrelated antibiotics.

2.2 RNA-Sequencing and Differential Gene Expression Analysis

RNA sequencing (RNA-seq) enables genome-wide, quantitative measurement of transcript abundance with a broad dynamic range and the ability to detect novel transcripts compared with microarrays. Typical RNA-seq analysis pipelines include quality control, read trimming, alignment to a reference genome, quantification of reads per gene, and statistical testing for differential expression. Differential expression tools such as DESeq2 model count data using negative binomial distributions.

2.3 Reproducibility Challenges in RNA-Seq Analysis

RNA-seq results can vary substantially across pipelines due to differences in software versions, reference annotations, parameter choices, and statistical methods, which may alter the set of genes reported as differentially expressed. Even when using the same raw data, small changes in tools or settings can propagate to biologically relevant discrepancies in downstream interpretation. To mitigate this, current best practices emphasize tracking code with Git, encapsulating environments using Docker or similar containers, and orchestrating workflows with systems such as Nextflow or Snakemake to ensure that analyses remain executable and auditable over time.

2.4 The Reprohackathon Approach

Reprohackathons are training-oriented hackathon events where participants attempt to reproduce published analyses using modern reproducibility technologies. These projects expose students to real-world obstacles such as missing methods, unavailable data, deprecated software, and limited computational resources, while also demonstrating how containerization and workflow systems can overcome such barriers.

In this project, the reprohackathon centers on reproducing the RNA-seq component of the Peyrusson et al. study, focusing on differential gene expression between intracellular *S. aureus* persisters exposed to oxacillin and control bacteria. The workflow relies on Docker for environment management, Nextflow for pipeline automation, and Git for systematic tracking of code and configuration changes, with the goal of producing a fully reproducible and shareable analysis.

3 Materials and Methods

3.1 Study Design and Data Sources

The original study deposited RNA-seq data in the Gene Expression Omnibus (GEO) under accession number GSE139659 (`https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE139659`). The experimental design compared intracellular persisters—bacteria recovered from J774 macrophages exposed to 50× MIC oxacillin for 24 hours—against control samples consisting of bacteria mixed with cell lysate from non-infected macrophages.

The original experimental protocol involved infecting J774 macrophages with *S. aureus* strain SH1000 expressing GFP from a tetracycline-inducible promoter at a multiplicity of infection of 100:1. Following phagocytosis, cells were exposed to high concentrations of oxacillin to induce a homogeneous population of persisters. Intact, viable persisters were isolated by fluorescence-activated cell sorting (FACS) by gating GFP-positive, propidium iodide-negative events, ensuring that only live, intracellular bacteria were collected for RNA extraction. Three biological replicates were generated for both persister and control conditions.

Figure 1 summarizes the complete computational workflow implemented to reproduce the RNA-seq analysis, from data download to statistical analysis in DESeq2.

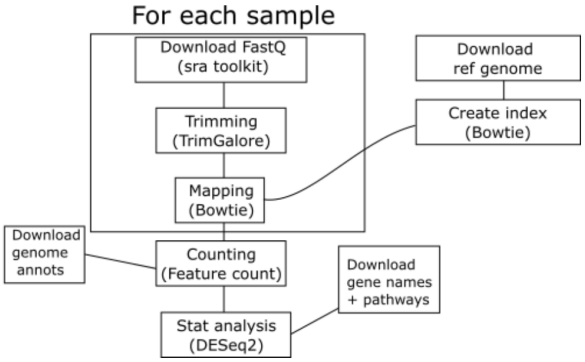


Figure 1: Complete Workflow

3.2 FASTQ File Download: Project-wide and Single-run Strategies

The initial step of our RNA-seq workflow involved the automated retrieval of raw sequencing data in FASTQ format for each SRA run accession associated with the study. To ensure computational reproducibility and ease of deployment across different environments, the download process was encapsulated both as a dedicated Nextflow process and within a custom Docker container built for the SRA Toolkit.

3.2.1 Unified Download Logic

All downloads are performed via the `DOWNLOAD_FASTQ` Nextflow process, which encapsulates the SRA Toolkit in a dedicated Docker container. This guarantees that every file, regardless of how it is selected, is downloaded in the same controlled, reproducible environment. The process not only downloads each FASTQ file, but also automatically compresses it (`.gz`) for efficient storage and downstream processing. Additionally, in "test mode," it can produce a truncated file containing only the first 10,000 reads for workflow validation.

Key script logic:

```
echo "Downloading␣FASTQ␣for␣${sra_id}"
prefetch ${sra_id}
if [ "${params.test}" == "true" ]; then
    # Test mode: keep first 10,000 reads
    fasterq-dump --threads ${task.cpus} -t . --progress ${sra_id}
    gzip ${sra_id}.fastq
    zcat ${sra_id}.fastq.gz | head -n 10000 | gzip > ${sra_id}_test.fastq.gz
    mv ${sra_id}_test.fastq.gz ${sra_id}.fastq.gz
else
    fasterq-dump --threads ${task.cpus} -t . --progress ${sra_id}
    gzip ${sra_id}.fastq
fi
echo "DOWNLOADING␣completed:␣${sra_id}"
```

This code downloads, compresses, and (if in test mode) truncates each FASTQ file.

3.2.2 Two Modes of SRR Retrieval

Download All Samples via SRA Project Accession: When the SRA project accession (e.g., SRP227811) is given: The `GET_SRR` process uses a Docker image with Entrez Direct installed to query the NCBI SRA database and retrieve all associated SRR numbers.

Example script:

```
esearch -db sra -query ${sra_project} | efetch -format runinfo | cut -d',' -f1 |
    grep SRR > SRR_list.txt
```

This extracts every SRR accession for the given project. The resulting list (e.g., six files: 3 persister, 3 control) is passed to the `DOWNLOAD_FASTQ` process. Each SRR is then downloaded using the same SRA Toolkit Docker, ensuring environment and method consistency.

Note: Entrez Direct Docker provides only the utilities (`esearch`, `efetch`, etc.) to extract all run accessions for a given project. It is not used for downloading FASTQ files themselves.

Download an Individual File via SRR: When a specific SRR number is provided (e.g., SRR10379726), the workflow directly invokes the `DOWNLOAD_FASTQ` process for that accession, using the SRA Toolkit Docker container.

3.2.3 Workflow Integration and Reproducibility

By separating SRR list generation (project context) from data download, and by using `DOWNLOAD_FASTQ` exclusively for all file retrieval, the workflow achieves:

- **Complete reproducibility** of the download operation through a single well-defined environment (SRA Toolkit Docker).
- **Flexibility** for users: download a whole project dataset (using SRA project accession + `GET_SRR`) or a single file (using one SRR).

- **Robustness and transparency**: only a single, well-audited process (`DOWNLOAD_FASTQ`) performs the key task of file downloading, with full provenance and containerization benefits for traceability, auditing, and scientific rigor.

3.3 Read Trimming: Trim Galore and Cutadapt

During this step, raw sequencing reads are processed to remove low-quality bases and adapter contamination before mapping. To mirror the pipeline from the original publication, we used Trim Galore version 0.4.4 and Cutadapt version 1.11, both packaged in a custom Docker image based on Ubuntu 18.04 and Python 3.6. This ensures our trimming environment is identical to that in the study, supporting strict reproducibility.

Each input FASTQ file is trimmed using a quality threshold of Q=20 (which keeps bases with $\geq 99\%$ base call accuracy) and sequences are retained only if they are at least 25 bases in length. The step also compresses the output for efficient storage and downstream use. The container setup verifies the installed version of Trim Galore to prevent discrepancies.

Trimming is automated in the workflow by the following Nextflow process:

```
# Remove .gz extension and .fastq extension to get SRA ID
SRA_ID=$(basename ${fastq_files} .fastq.gz)

trim_galore -q 20 --phred33 --length 25 ${SRA_ID}.fastq.gz

# Rename output file
# mv ${SRA_ID}.fastq.gz_trimmed.fq.gz ${SRA_ID}_trimmed.fq.gz
```

This script extracts the SRA ID from each filename and processes it with the specified parameters, ensuring that downstream alignment and quantification are performed on high-quality, adapter-free reads. By replicating both the software versions and the precise logic of the original workflow, we maintain analytical fidelity and reliability throughout this step.

3.4 Reference Genome and Annotation Download

To replicate the RNA-seq analysis pipeline exactly, the reference genome and its genome annotation for *Staphylococcus aureus* were retrieved from NCBI using the accession number CP000253.1. This step was automated in the reproducible workflow using a dedicated Nextflow process, `GET_DATA_GENOME`, which interfaces with the NCBI Entrez system for nucleotide data retrieval.

The process downloads two essential files: the reference genome sequence in FASTA format reference.fasta, and the genome annotation in GFF3 format (`annotation.gff`). Both files serve as critical inputs for mapping and feature quantification steps downstream.

This process uses the `esearch` and `efetch` utilities provided by the Entrez Direct Docker container to ensure consistency and reproducibility. The container guarantees that the exact same versions of Entrez tools are used, avoiding variability in database queries.

Process script excerpt:

```
esearch -db nucleotide -query $ref_genome | efetch -format fasta > reference.
    fasta
esearch -db nuccore -query $ref_genome | efetch -format gff3 > annotation.gff
```

Here, `$ref_genome` corresponds to the NCBI accession CP000253.1. This two-step command first queries the nucleotide database to download the genome FASTA sequence and then queries the nucleotide core database for the corresponding annotation in GFF3 format.

Providing an automated, containerized environment for this step avoids manual downloads, ensures file versions match those used in the original analysis, and supports full reproducibility within the workflow.

3.5 Index creation using Bowtie

The next step is to index the reference genome. Without an index, the algorithm would have to linearly scan the entire genome for each read, which would be extremely slow. The index allows the data to be structured for efficient searching. The library used to perform this indexing is Bowtie. For reproducibility purposes, we work on a docker containing Bowtie version 0.12.7. In the following process, we will also use samtools and bowtie. So we create a single "bowtie" docker that contains both libraries with

the appropriate versions and the latest ubuntu version. The docker image is stored on dockerhub at the following address: mariemeier/reprohackathon:bowtie. We create a new nextflow process named `INDEX_REF_GENOME` that takes the reference fastq file as input and returns the indexed reference genome in the results/index directory. Indexing is performed by the bowtie-build function within the docker. The next bash command is executed :

```
mkdir -p index_ref_genome
bowtie-build ${ref_genome} index_ref_genome/indexed_ref_genome
```

The resulting file is an .ebwt file based on the Burrows-Wheeler transform (BWT) of the reference genome. It contains the transformed sequence of the genome, the metadata needed to reconstruct the positions of the patterns in the original genome, and optimized data structures for fast sequence searching.

3.6 Mapping using Bowtie

The next process involves mapping the genome being studied to the indexed reference genome. Mapping is aligning short sequences (reads) obtained from sequencing with a reference sequence, which allows us to determine where they are located. The process is executed in the same "bowtie" docker as before. The new nextflow `MAPPING_BOWTIE` process generates a .bam file containing the alignments. The process uses 3 CPU cores for execution. The results are copied to the results/mapping folder, overwriting existing files. The process takes as input a tuple containing `trimmed_fastq` (path to the compressed fastq file `*_trimmed.fq.gz` containing the trimmed reads) and `indexed_genome` (path to the folder containing the reference genome index). The process generates a .bam file for each input FASTQ file. This is a binary (compressed) alignment format that contains the reads aligned to the reference genome. It is created from the .sam file from bowtie using samtools according to the following script:

```
gunzip -c ${trimmed_fastq} | \
bowtie -S ./index_ref_genome/indexed_ref_genome - | \
samtools view -@ ${task.cpus} -bS -o ${srr_id}.bam
```

3.7 Feature Count

This step consists of creating a count matrix from the mapped sequence file. This involves counting the number of reads aligned with each gene between the two conditions. We are working on a docker containing SubReads version 1.4.6-p3 available here: https://sourceforge.net/projects/subread/. It requires Ubuntu version 22.04 to use the featureCounts binary. The process named FEATURECOUNTS takes the .bam file of mapped reads as input and the gff file containing the genes location for the reference genome obtained from the process `GET_DATA_GENOME`. The output is a count matrix, where rows represent genes and columns represent samples, and a summary file is generated for each sample, detailing statistics such as the number of assigned and unassigned reads. The next script is executed :

```
featureCounts -t gene -g ID -s 1
-a ${gff_file} -o counts.txt ${mapped_reads}
```

3.8 Creation of ColData file

In order to know which SRR corresponds to which condition, we create a tsv file containing two columns, the first being `ID_SRR` and the second being the condition (IP or control). To do this, the `GET_GEO_TABLE` process downloads and extracts metadata from the NCBI's GEO (Gene Expression Omnibus) database. It generates a table file (TSV) that associates sample identifiers (GSM) with their respective experimental conditions (e.g., "IP" or 'control'). It takes `geo_id` as input and returns the corresponding tsv file. The following script is executed in the "entrez-direct" docker:

```
geo_parent=\$(echo ${geo_id} | sed -E 's/(GSE[0-9]{3})[0-9]
+/\\1nnn/')
wget -q ftp://ftp.ncbi.nlm.nih.gov/geo/series/\${geo_parent}
/${geo_id}/soft/${geo_id}_family.soft.gz -O ${geo_id}.soft.gz
gunzip -f ${geo_id}.soft.gz
awk '
/^\\^SAMPLE/ {gsm=\$3}
```

```
/^!Sample_description/ {
    desc=\$3
    if (desc ~ /^IP/) cond="IP"
    else if (desc ~ /^ctrl/) cond="control"
    else cond="unknown"
    print gsm"\\t"cond
}' ${geo_id}.soft > ${geo_id}_table.tsv
```

Furthermore, the `GET_SRA_DATA` process takes an SRP value as input and returns a CSV file containing the metadata for the SRA runs associated with the project. The following script performs this task and is executed in the container "entrez-direct":

```
esearch -db sra -query "${sra_project}"
| efetch -format runinfo > "sra_data_complete.csv"
```

Then the `CREATE_COLDATA` process uses a R script to merge the GEO and SRA metadata and generate a structured colData file. It takes as input a tuple containing the paths to `geo_id`, `sra_data_file`, and the R script, and returns the structured tsv coldata file for further analysis. It is executed in the deseq2 container at the following address: mariemeier/reprohackathon:deseq2. It contains R version 3.4.1, CRAN packages, Biocinstaller 3.5 to obtain the bioconductor packages, Rgraphviz version 2.54.0, and DESeq2 version 1.16.1. The script is :

```
Rscript ${script_R} ${geo_id} ${sra_data_file} coldata.tsv
```

3.9 Statistical Analysis (DESeq2)

The last process performs statistical analysis in order to reproduce the two genome comparison graphs. Ran in the container deseq2, the nextflow process `STAT_ANALYSIS` takes as input a tuple with the paths of coldata, `count_matrix`, and the R script and returns the plot according to the following script:

```
Rscript ${script_R} ${coldata} ${count_matrix} ma_plot.png
```

3.10 Comparing results with the original study

To quantitatively assess the concordance between our differential expression results and those reported by Peyrusson et al., we implemented a dedicated Nextflow process, `RESULTS_COMPARISON`. This process utilizes an R script to compare $\log_2$ fold changes from our DESeq2 analysis with the reference values provided in the original publication. The process takes as input a tuple containing the paths to our DESeq2 results file, the reference results file from Peyrusson et al., and the R script for comparison. It outputs a scatter plot visualizing the correlation between the two sets of $\log_2$ fold changes, along with a computed Pearson correlation coefficient.

```
Rscript ${script_R} ${matrix_stat} ${results_article} comparison_plot.png
```

The paper also contains a plot showing only the translation genes that we have to reproduce. In order to exctract translation genes, we had to search for Kegg orthology numbers corresponding to translation genes. We found them on the Kegg website (`https://www.kegg.jp/brite/ko03012`). The script filters the DESeq2 results to retain only genes with KOs in the provided list, and generates a plot similar to the MA-plot but restricted to translation genes. We used the following saos : sao03010 (Ribosome, factors, etc.), sao03012 (Aminoacyl-tRNA biosynthesis), sao00970 (Amino Acid Metabolism) and sao01613 (Aminoacyl-tRNA synthetases). Concerning the latter, we had tu enumerate every KO identifiers corresponding to aminoacyl-tRNA synthetases.

4 Results

4.1 Count Matrix

The read-counting step using `featureCounts` (v1.4.6-p3) produced a gene-level count matrix for the six RNA-seq samples, corresponding to the three persister and three control conditions. The command was run in strand-specific mode (`-s 1`) on features annotated as genes in `annotation.gff` (`-t gene`,

`-g ID`), and took as input the six BAM files generated by the Bowtie mapping step (`SRR10379725.bam`, `SRR10379724.bam`, `SRR10379723.bam`, `SRR10379722.bam`, `SRR10379721.bam`, `SRR10379726.bam`).

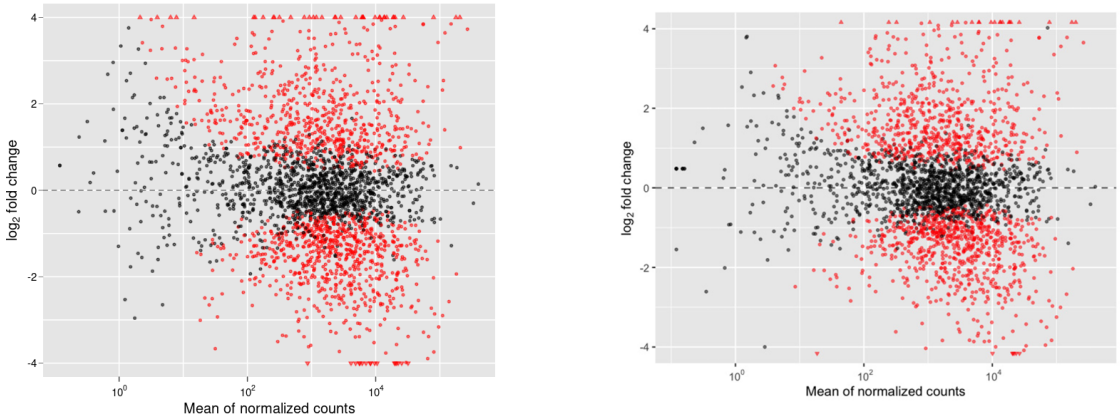
The resulting `counts.txt` file is organized with one header line containing genomic coordinates and one column per sample, followed by one row per gene such as `gene-SAOUHSC_00001`, `gene-SAOUHSC_00002`, etc. For each gene, the matrix reports its chromosome (CP000253.1), start and end positions, strand, length, and the raw integer counts in the six samples, for example very highly expressed genes such as `gene-SAOUHSC_00009` (up to tens of thousands of reads) and more weakly expressed genes with only a few reads. This structure matches the expected gene-by-sample input format for downstream normalization and differential expression analysis with DESeq2.

4.2 Differential Expression and Reproduction of the MA-plot

Using the `STAT_ANALYSIS.R` script, the DESeq2 analysis was executed directly from the count matrix and `coldata` file to reproduce the MA-plot of the complete RNA-seq dataset shown in Supplementary Figure 3 of Peyrusson *et al.*. The script builds a `DESeqDataSet` with design `~ Condition`, runs the standard DESeq2 pipeline with default parameters, and extracts the contrast *IP vs. control*, yielding for each gene a base mean of normalized counts, a $\log_2$ fold change, and an adjusted p-value.

The PCA of the Variance Stabilizing Transformation (VST) data was performed to assess global sample variance (Supplementary Fig.1). The plot clearly shows that the Condition variable (IP vs. Control) accounts for the majority of the variance ($PC1$: 86%), demonstrating a strong separation between the biological groups. However, one control sample (SRR10379726) is identified as a clear outlier, lying significantly distant from its two biological replicates. This indicates a major technical or biological deviation in that specific sample, which may have affected the dispersion estimation in the control group.

From these results, we computed the MA coordinates (A = $\log_2(baseMean)$, M=$\log_2$ fold change) and generated a custom base-R plot, exporting the figure as `ma_plot.png` (Figure 2.a).



(a) MA-plot from our DESeq2 analysis.

(b) MA-plot from the article.

Figure 2: Comparison of MA-plots: (a) reproduced from our DESeq2 analysis; (b) original from Peyrusson *et al.* Supplementary Figure 3. Red points indicate genes with adjusted p-value < 0.05, black points non-significant genes.

Our MA-plot reproduces both the global structure and the visual style of the original supplementary figure: genes are centred around a $\log_2$ fold change of zero, with increased dispersion at low counts and a clear band of significant genes (red points) extending towards positive and negative fold changes at higher mean expression values. As in the publication, $\log_2$ fold changes are truncated at ± 4, with points beyond these thresholds shown as triangles at the top or bottom of the plot, and significance is highlighted with an adjusted p-value cutoff of 0.05. Qualitatively, the density and spread of significant genes across the dynamic range of normalized counts are very similar to those reported by Peyrusson *et al.*, indicating

that our workflow rapidly recovers a transcriptomic signature that is globally consistent with the original RNA-seq differential expression analysis.

4.3 Correlation of Log_2 Fold Changes and padj with the Original Study

To quantitatively assess how well our differential expression analysis reproduces the results of Peyrusson *et al.*, we compared our DESeq2 $\log_2$ fold changes with the reference $\log_2$ fold changes provided by the authors for each gene. For this purpose, we implemented the script `RESULTS_COMPARISON.R`, which reads our statistical results and the reference results, joins them by gene identifier (`Locus_Tag`), and computes a Pearson correlation coefficient between the two sets of $\log_2$ fold changes. Genes are then classified into four categories according to their adjusted p-values in both analyses: significant in both datasets, significant only in our data, significant only in the article, or non-significant in either.

The script produces a scatter plot where each point represents a gene, with the reference $\log_2$ fold change on the x-axis and our calculated $\log_2$ fold change on the y-axis (Figure 3). The points closely cluster around the diagonal, and the overall Pearson correlation reaches $R = 0.915$, indicating a strong concordance between our pipeline and the original analysis. Genes that are significant in both datasets (dark red points) align particularly well with the diagonal, while genes significant in only one of the two analyses (orange or dark blue points) are more dispersed, highlighting the impact of borderline p-values, multiple-testing corrections, and small differences in preprocessing or software versions.

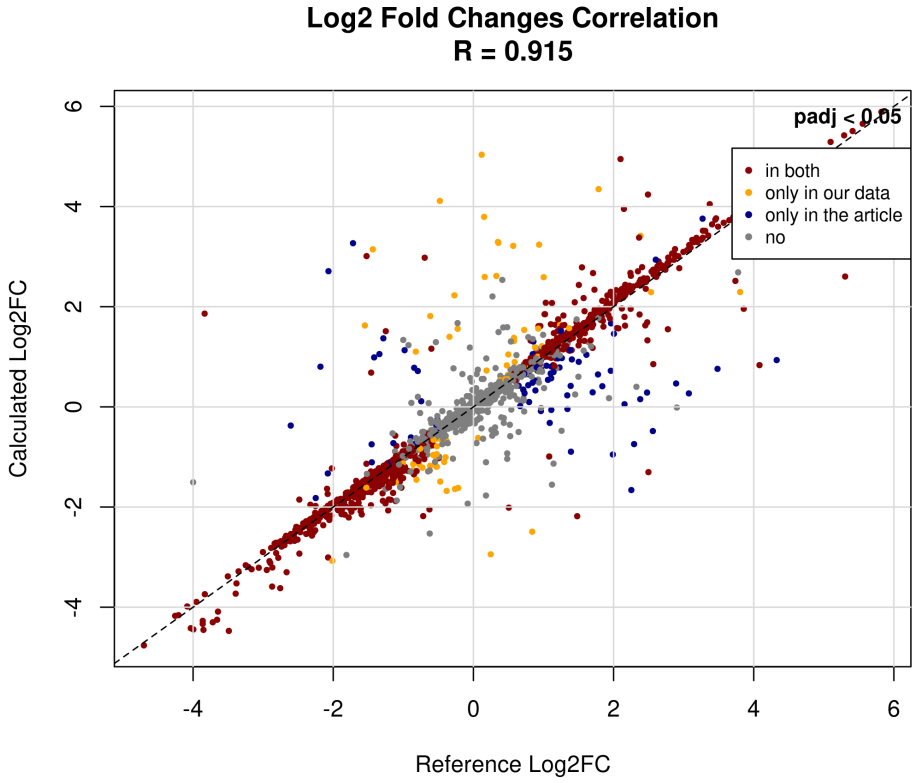
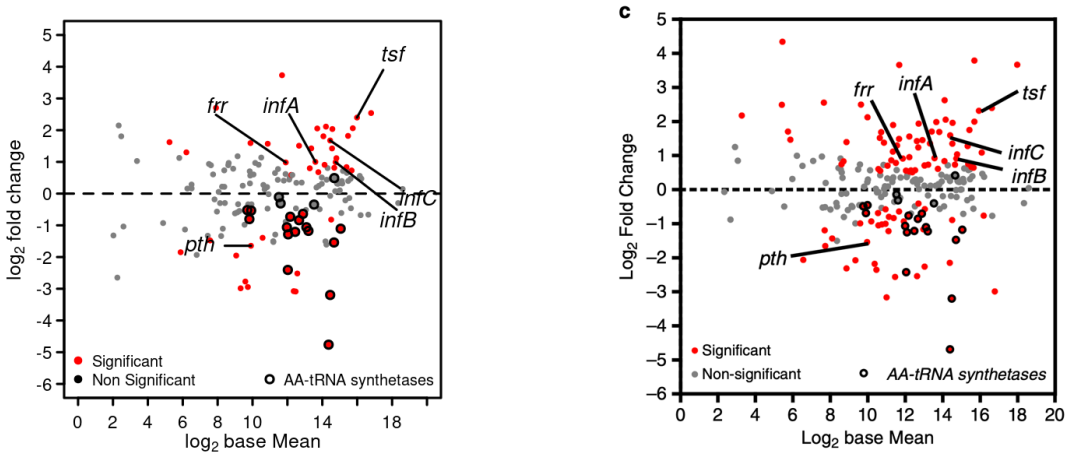


Figure 3: Correlation between reference $\log_2$ fold changes from Peyrusson *et al.* and $\log_2$ fold changes recalculated with our DESeq2 workflow. Each point corresponds to one gene; colours indicate significance categories (padj < 0.05) in the two analyses.

4.4 Reproduction of the plot of translation genes

After filtering the genes, the script `STAT_ANALYSIS.R` returns the plots with only translation-related genes, as shown in Figure 4a below.



(a) MA-plot from our DESeq2 analysis.

(b) MA-plot from the article.

Figure 4: Comparison of MA-plots for translation-related genes: (a) reproduced from our DESeq2 analysis; (b) original from Peyrusson *et al.* Figure 3b. Red points indicate genes with adjusted p-value < 0.05, black points non-significant genes.

The plots are quitte similar exept that in our plot, there are less significant genes (in red) than in the article's plot. This may be due to differences in the SAO and KO numbers used to filter the genes, as KEGG certainly updates their databases since the publication of the article. It may also be explained by small differences in the preprocessing steps or software versions used in our workflow compared to the original analysis.

5 Discussion

This project presented a great opportunity to confront both the practical and theoretical challenges of reproducing published RNA-seq analyzes in a fully transparent and reproducible manner. Among the key lessons learned was the care that needed to be taken in selecting, installing and documenting the correct versions of all libraries and tools. The creation of Docker containerized environments for each main component of the analyzes, such as the SRA Toolkit, Bowtie, and Samtools, ensured computational reproducibility and avoided a series of common problems related to software incompatibilities and version changes. Different values of the parameters in Trim Galore have been tested out (for example plots with q = 30 or length = 50), but the results (Supplementary Fig. 2,3) did not change much. We could have tested reproducing both plots by using newer containers for each process, to test the limits of reproducibility when the software versions are different from the original article.

The time consumed by researching and checking compatible library versions was not superfluous, this was quite important at each step of the workflow execution and for supporting collaboration within the team. Indexing the reference genome and mapping the reads of the sequence to it, using Nextflow workflows and Dockerized Bowtie, showed how crucial accurate sequence alignment is for later gene expression analyzes. Thanks to careful workflow management and strict control of the computational environment, the mapping steps produced reliable alignment results, forming a solid basis for differential expression analysis. This reproducible workflow made it possible to correctly identify and extract sequences of interest, specifically the differentially expressed genes typical of intracellular *S. aureus* persisters exposed to antibiotics.

This project highlights the increasing importance of automation and reproducibility frameworks in computational biology. Future work should aim to further improve workflow modularity and scalability, extend the pipeline to support more experimental situations, and integrate new analytical tools as they become available. Finally, keeping a strong focus on transparency, documentation, and version control will remain essential to ensure that bioinformatics analyzes stay robust, shareable, and accessible to the wider scientific community.

6 References

1. Peyrusson, F., Varet, H., Nguyen, T. K., Legendre, R., Sismeiro, O., Coppe, J. Y., Leduc, D. (2020). Intracellular *Staphylococcus aureus* persisters upon antibiotic exposure. *Nature Communications*, 11, 2200. https://doi.org/10.1038/s41467-020-15966-7

2. Langmead, B., Trapnell, C., Pop, M., Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3), R25. https://doi.org/10.1186/gb-2009-10-3-r25

3. Love, M. I., Huber, W., Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12), 550. https://doi.org/10.1186/s13059-014-0550-8

4. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., ... Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16), 2078–2079. https://doi.org/10.1093/bioinformatics/btp352

5. Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4), 316–319. https://doi.org/10.1038/nbt.3820

6. Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014(239), 2.

7. Ewels, P., Magnusson, M., Lundin, S., Käller, M. (2016). MultiQC: Summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, 32(19), 3047–3048. https://doi.org/10.1093/bioinformatics/btw354

8. Bolger, A. M., Lohse, M., Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina sequence data. *Bioinformatics*, 30(15), 2114–2120. https://doi.org/10.1093/bioinformatics/btu170

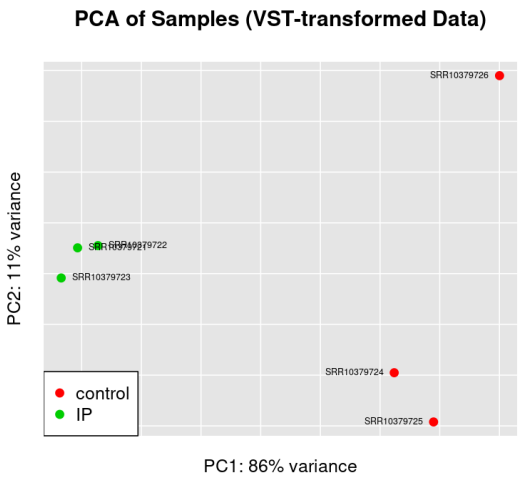
9. Liao, Y., Smyth, G. K., Shi, W. (2014). featureCounts: An efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 30(7), 923–930. https://doi.org/10.1093/bioinformatics/btt656

10. Kodama, Y., Shumway, M., Leinonen, R. (2012). The Sequence Read Archive: Rapid sharing of data and metadata related to sequence submissions. *Nucleic Acids Research*, 40(D1), D54–D56. https://doi.org/10.1093/nar/gkr854

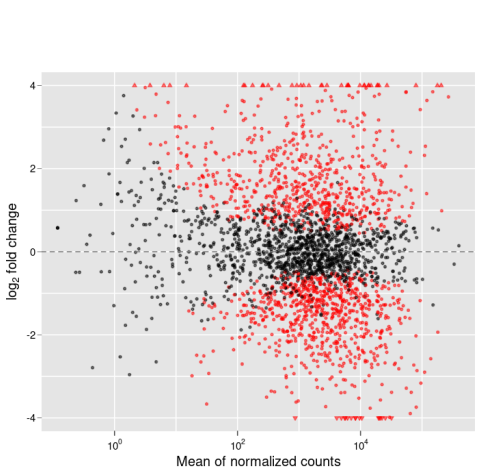
11. Git - Version Control System. Available at https://git-scm.com Accessed November 2025.

12. Docker Documentation. Available at https://docs.docker.com Accessed November 2025.

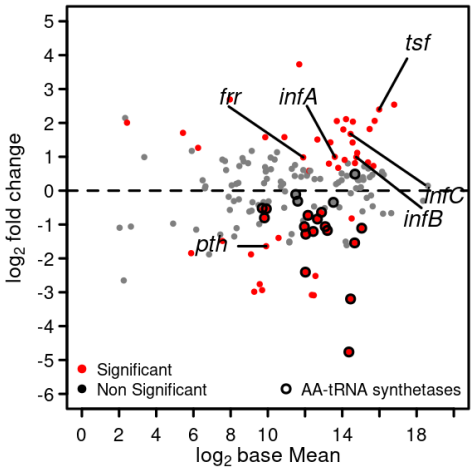
Supplementary Figures



Supplementary Figure 1. Principal Component Analysis (PCA) of VST-transformed samples. The plot validates separation based on biological condition (IP vs. Control) and identifies sample outliers.

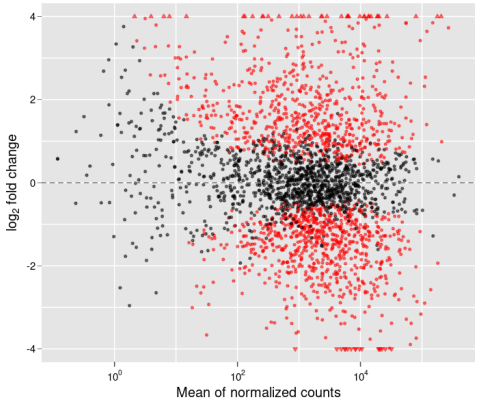


(a) Global MA-plot of results obtained using a strict quality filter (Q30 vs. Q20 baseline).

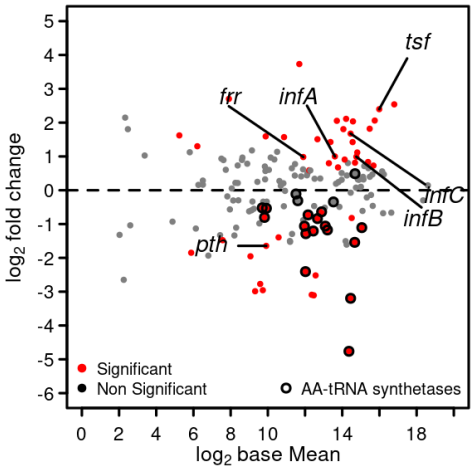


(b) MA-plot of Translation-related genes using a strict quality filter (Q30).

Supplementary Figure 2: Impact of Strict Quality Filtering (Q30). Comparison of global differential expression and translation gene enrichment.



(a) MA-plot with Length = 50 nt (Strict Length Filter).



(b) MA-plot of Translation Genes with length = 50 nt.

Supplementary Figure 3: Impact of Minimum Read Length (50 nt). Comparison of differential expression results with the original protocol (25 nt baseline).