

WPI

BIOINFORMATICS & COMPUTATIONAL BIOLOGY PROGRAM

Introduction to RNA-seq Data Processing and Differential Gene Expression Analysis with *Pseudomonas putida*

Gabrielle R. CABEBE

Gcabebe@wpi.edu

Professor:

Dr. Natalie G. FARNY

May 3, 2024

TABLE OF CONTENTS

1. Introduction	2
2. Before Getting Started...	2
2.1. Command Line & Turing Basics	2
2.2. Managing Environments with Conda	2
2.3. Installing Packages	3
2.3.1. Installing Packages with Conda	3
2.3.2. Installing Packages from Download Links	3
3. Browsing Datasets	3
3.1. Understanding the Source of Your Data	4
3.1.1. ENA (European Nucleotide Archive)	4
3.1.2. NCBI SRA	5
4. Obtaining your Raw Data	5
4.1. The ENA Download Script	5
4.1.1. Single Projects	5
4.1.2. Multiple Projects at Once	7
5. Quality Control with FastQC	8
6. Trimming Data and Removing Adapters	9
7. Mapping Reads with Alignment	10
7.1. Download Genomic Data	10
7.2. STAR Indexing	12
7.3. STAR Alignment	13
8. Generate Read Count Table	15
9. Differential Gene Expression Analysis (DGE)	16
References	17

1. Introduction

Ever since its introduction in 2008, RNA sequencing (RNA-seq) has become an indispensable tool for transcriptome analysis [1]. With the copious amount of RNA-seq data being generated, a new challenge of extracting the most meaning out of this data with as little misinterpretation as possible. This data extraction requires appropriate background and computational skills, as well as the necessary tools to carry this out. More thorough documentation for RNA-seq processing is available for the most studied organisms, such as *E. coli* and human tissues. However, these resources may be more difficult to locate for lesser known organisms. The goals of this workflow is to break down each step of the RNA-seq data processing using the prokaryotic organism *Pseudomonas putida*.

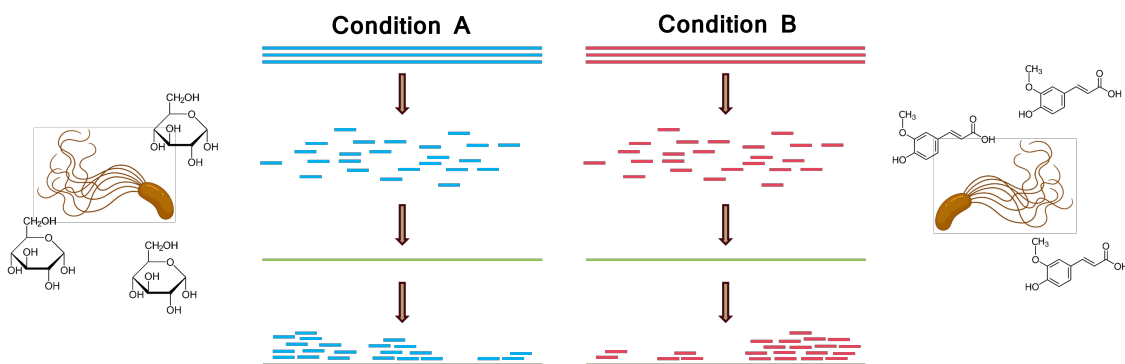


Fig. 1. RNA-seq - *In vivo*, *In vitro*, to *In silico*

2. Before Getting Started...

2.1 Command Line & Turing Basics

This guide requires heavy usage of the command line with Turing, a high performance computing research cluster. A basic user guide for Turing can be found on the [WPI Academic & Research Computing webpage](#). It includes how to login to Turing, prepare and running a job, utilizing academic and research software modules.

2.2 Managing Environments with Conda

Throughout preprocessing on Turing, you will need to install several packages to go through the pipeline. However, some packages may conflict with each other due to different versions of these packages or with Python. For example, one tool may require Python 3.7.0, while another tool may need Python 3.8.0. Instead of installing Python 3.7.0, uninstalling that version, and then installing Python 3.8.0, it is better to work with 2 different environments that have each required Python version.

In order to avoid this issue, it is best to work with different environments that each

have a different Python version. Conda is a popular command line tool for managing different environments, as well as installing packages which will be discussed in the following section.

[Conda documentation](#) is available on their website for steps installing and creating different environments.

2.3 Installing Packages

At each step of this analysis, the person utilizing this guide is expected to download each tool to their turing account. There are 2 ways to do this.

2.3.1 Installing Packages with Conda

Several of the tools throughout this workflow can be downloaded with conda with the following command:

```
1 conda install [package_name]
```

2.3.2 Installing Packages from Download Links

```
1 wget [download_link]
```

From there, configure the `.bashrc` file, a configuration file for the Bash shell, to allow easier access to each tool as opposed to running the absolute file path.

We will use the tool FastQC as an example. Without configuration of the `.bashrc` file, the command would be the following:

```
1 /home/gcabebe/~/fastqc [sample.fastqc.gz]
```

To configure your `.bashrc` file, go to the start directory (eg. `/home/gcabebe`) and enter `vim .bashrc` to enter the viewing and editing space for this file.

Anywhere below `module load slurm`, enter the following:

```
1 export PATH="$PATH:/home/gcabebe/./FastQC"
```

We will now have the following:

```
1 fastqc [sample.fastqc.gz]
```

3. Browsing Datasets

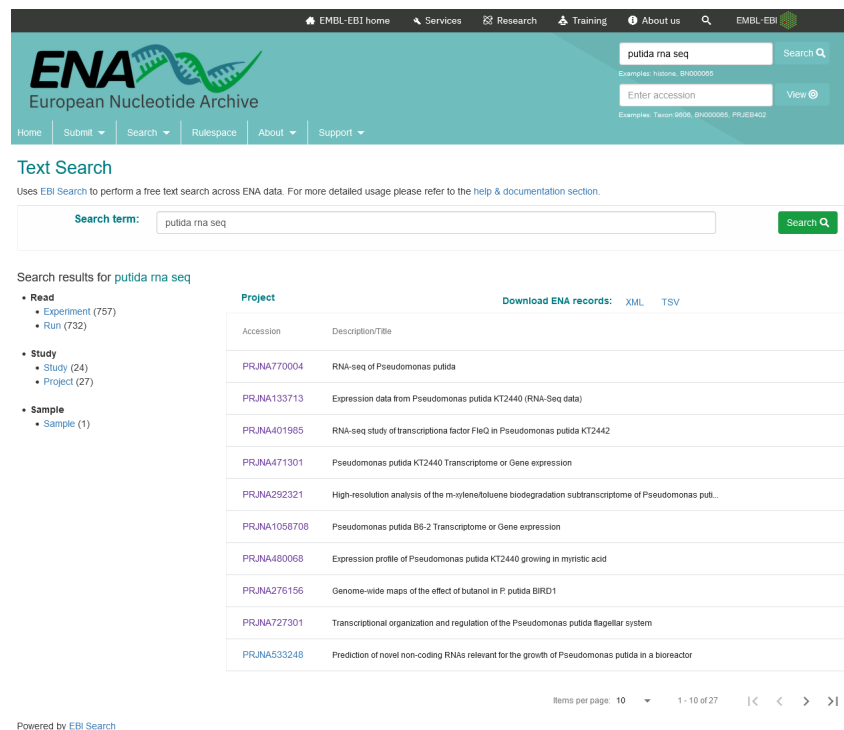
You may have data that has been generated from your lab that you're analyzing instead. In that case, you may skip this section. Otherwise, refer to this section on downloading publicly available data.

3.1 Understanding the Source of Your Data

3.1.1 ENA (European Nucleotide Archive)

The European Nucleotide Archive (ENA) is an open platform for managing, sharing, and disseminating sequence data. This is the database to freely download RNA-seq data in .fastq.gz format. It's not as thorough as the NCBI database, but ENA is a good starting place to give you a brief description of RNA-seq data related to your organism of interest.

In the top right of the ENA homepage, there are 2 search boxes: one for searching by key terms, and another for searching by accession number. By simply putting 'putida rna seq', and organizing results by project, we can curate a list of studies with a brief description for each project (Fig. 2).



ENA
European Nucleotide Archive

Search term: putida rna seq

Search results for putida rna seq

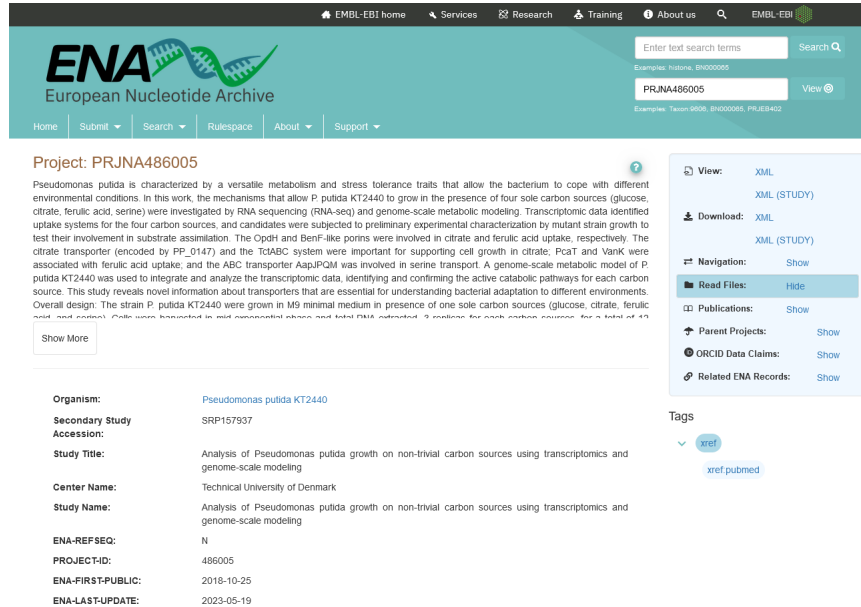
- Read
 - Experiment (757)
 - Run (732)
- Study
 - Study (24)
 - Project (27)
- Sample
 - Sample (1)

Accession	Description/Title
PRJNA770004	RNA-seq of Pseudomonas putida
PRJNA133713	Expression data from Pseudomonas putida KT2440 (RNA-Seq data)
PRJNA401985	RNA-seq study of transcription factor FleQ in Pseudomonas putida KT2442
PRJNA471301	Pseudomonas putida KT2440 Transcriptome or Gene expression
PRJNA292321	High-resolution analysis of the m-xylene/toluene biodegradation subtranscriptome of Pseudomonas putida...
PRJNA1058708	Pseudomonas putida B6-2 Transcriptome or Gene expression
PRJNA480068	Expression profile of Pseudomonas putida KT2440 growing in myristic acid
PRJNA276156	Genome-wide maps of the effect of butanol in P. putida BIRD1
PRJNA727301	Transcriptional organization and regulation of the Pseudomonas putida flagellar system
PRJNA533248	Prediction of novel non-coding RNAs relevant for the growth of Pseudomonas putida in a bioreactor

Items per page: 10 1 - 10 of 27

Fig. 2. ENA Project Search Results

We will work with project PRJNA486005, which analyzes the mechanisms of *P. putida* while growing in 4 different carbon sources: glucose, citrate, ferulic acid, and serine. When you click on the link on this accession number, it will bring you to a page with more information on this study (Fig. 3).



Project: PRJNA486005

Pseudomonas putida is characterized by a versatile metabolism and stress tolerance traits that allow the bacterium to cope with different environmental conditions. In this work, the mechanisms that allow *P. putida* KT2440 to grow in the presence of four sole carbon sources (glucose, citrate, ferulic acid, serine) were investigated by RNA sequencing (RNA-seq) and genome-scale metabolic modeling. Transcriptomic data identified uptake systems for the four carbon sources, and candidates were subjected to preliminary experimental characterization by mutant strain growth to test their involvement in substrate assimilation. The OpiH and BenF-like porins were involved in citrate and ferulic acid uptake, respectively. The citrate transporter (encoded by PP_0147) and the TcABC system were important for supporting cell growth in citrate. PcaT and VanK were associated with ferulic acid uptake, and the ABC transporter AapJPGM was involved in serine transport. A genome-scale metabolic model of *P. putida* KT2440 was used to integrate and analyze the transcriptomic data, identifying and confirming the active catabolic pathways for each carbon source. This study reveals novel information about transporters that are essential for understanding bacterial adaptation to different environments. Overall design: The strain *P. putida* KT2440 were grown in M9 minimal medium in presence of one sole carbon sources (glucose, citrate, ferulic acid, and serine). Cells were harvested in mid-exponential phase and total RNA extracted. 2 samples for each carbon source, for a total of 8 samples.

[Show More](#)

Organism:	Pseudomonas putida KT2440
Secondary Study Accession:	SRP157937
Study Title:	Analysis of <i>Pseudomonas putida</i> growth on non-trivial carbon sources using transcriptomics and genome-scale modeling
Center Name:	Technical University of Denmark
Study Name:	Analysis of <i>Pseudomonas putida</i> growth on non-trivial carbon sources using transcriptomics and genome-scale modeling
ENA-REFSEQ:	N
PROJECT-ID:	486005
ENA-FIRST-PUBLIC:	2018-10-25
ENA-LAST-UPDATE:	2023-05-19

View: XML
XML (STUDY)

Download: XML
XML (STUDY)

Navigation: Show

Read Files: Hide

Publications: Show

Parent Projects: Show

ORCID Data Claims: Show

Related ENA Records: Show

Tags

xref
xref pubmed

Fig. 3. ENA Project Description

3.1.2 NCBI SRA

SRA (Sequence Read Archive) data is a database hosted by NCBI servers that serves as the largest publicly available repository of high throughput sequencing data. SRA sequences can be downloaded from Entrez search results, but this guide will focus on downloading data with ENA.

Refer to the [SRA download guide](#) on the NCBI website for more details on tools installation and commands to run.

4. Obtaining your Raw Data

4.1 The ENA Download Script

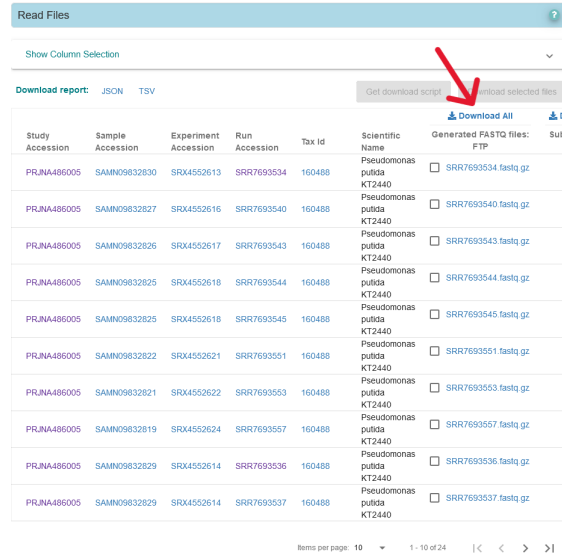
4.1.1 Single Projects

Starting from the project description page for PRJNA486005, scrolling to the bottom will show you a table with several accession numbers, name of the organism, and a download link for each sample in .fastq.gz format. For downloading the first sample in this project, for example, you can run the following line:

```
1 wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/
   ↪ SRR769/004/SRR7693534/SRR7693534.fastq.gz
```

wget is the non-interactive network downloader. It is a bash command used to download a file, in this case our single sample, from the link that follows the command.

In order to download all samples in this project at once, click on the 'Download All' link to download a script that will be used to run on Turing (Fig. 4).



Study Accession	Sample Accession	Experiment Accession	Run Accession	Tax id	Scientific Name	Generated FASTQ files:	Subi
PRJNA486005	SAMN09832830	SRX4552613	SRR7693534	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693534.fastq.gz	
PRJNA486005	SAMN09832827	SRX4552616	SRR7693540	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693540.fastq.gz	
PRJNA486005	SAMN09832826	SRX4552617	SRR7693543	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693543.fastq.gz	
PRJNA486005	SAMN09832825	SRX4552618	SRR7693544	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693544.fastq.gz	
PRJNA486005	SAMN09832825	SRX4552618	SRR7693545	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693545.fastq.gz	
PRJNA486005	SAMN09832822	SRX4552621	SRR7693551	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693551.fastq.gz	
PRJNA486005	SAMN09832821	SRX4552622	SRR7693553	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693553.fastq.gz	
PRJNA486005	SAMN09832819	SRX4552624	SRR7693557	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693557.fastq.gz	
PRJNA486005	SAMN09832829	SRX4552614	SRR7693536	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693536.fastq.gz	
PRJNA486005	SAMN09832829	SRX4552614	SRR7693537	160488	Pseudomonas putida KT2440	<input type="checkbox"/> SRR7693537.fastq.gz	

Fig. 4. ENA Downloadable Files

This script utilizes `wget -nc`, where `-nc` refers to `--no-clobber`. If the same file is downloaded twice in the same directory, the original file is preserved and `wget` will not download a new copy of the same file. explainshell.com gives a thorough explanation of these and other commands, and what combinations of commands may give you.

In order to implement this script on Turing, it must be submitted as a SLURM script. Below is an example of downloading the first 4 samples for project PRJNA486005:

```

1 #!/bin/bash
2 #SBATCH -N 1
3 #SBATCH -n 4
4
5 wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR769/006/SRR7693536/
   ↪ SRR7693536.fastq.gz
6 wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR769/003/SRR7693553/
   ↪ SRR7693553.fastq.gz
7 wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR769/005/SRR7693555/
   ↪ SRR7693555.fastq.gz
8 wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR769/009/SRR7693539/
   ↪ SRR7693539.fastq.gz

```

Each of the fields at the top of the script indicates the following:

1. `#!/bin/bash` is called a shebang. It tells the shell what program to interpret the

script with when executed. In this case, the script will be interpreted and run by the bash shell.

2. #SBATCH -N 1 requests 1 node for the job
3. #SBATCH -n 4 requests 4 tasks (CPU cores) per node

There are other fields you can indicate to request how much memory is needed, name of the job, time needed for the job, etc. Refer to WPI ARC's [Turing Basic User Guide](#) for more options.

Let's say the name of your file is 'rnaseq_download.sh'. To run this job on Turing, use the sbatch command:

```
1 sbatch rnaseq_download.sh
```

After you submit this job, a 'slurm-###.out' file will be created. You can keep track of what stage your job is at by viewing this file. You can also view how long your job has been running with:

```
1 queue -u wpiusername
```

When your job is finished, all downloaded FASTQ data will be stored in the same directory you ran the script in.

4.1.2 Multiple Projects at Once

In order to download datasets from more than one study at a time, the Listing 2 script can be used. This utilizes the [ENA File Downloader](#) []. View the documentation on the GitHub page for this tool for instructions on installing JDK8 and using the Gradle wrapper.

As an example, let's say we wanted to analyze all RNA-seq data on *P. putida* strain KT2440 growth on LB media in various conditions. On the search results page organized by Project, there is a link at the top of the table to download the search results as a TSV:

accession	description
"PRJNA455892"	"Pseudomonas putida KT2440 transcriptome - relA_LB_2"
"PRJNA455891"	"Pseudomonas putida KT2440 transcriptome - relA_LB_1"
"PRJNA455893"	"Pseudomonas putida KT2440 transcriptome - relA_LB_3"
"PRJNA455888"	"Pseudomonas putida KT2440 transcriptome - WT_LB_1"
"PRJNA455890"	"Pseudomonas putida KT2440 transcriptome - WT_LB_3"
"PRJNA455889"	"Pseudomonas putida KT2440 transcriptome - WT_LB_2"
"PRJNA520373"	"Pseudomonas putida KT2440 JBEI-13809 transcriptome - seedKTc"
"PRJNA520372"	"Pseudomonas putida KT2440 JBEI-13809 transcriptome - seedKTb"
"PRJNA520371"	"Pseudomonas putida KT2440 JBEI-13809 transcriptome - seedKTa"

With the ENA File Downloader and a TSV of projects for downloading our samples of interest, we can now run the following script:


```
1 #!/bin/bash
2 #SBATCH -N 1
3 #SBATCH -n 4
4 #SBATCH --mem=0
5 #SBATCH -p long
6 #SBATCH -t 30:00:00
7
8 ROOT_PATH='~/rnaseq'
9
10 # accessions downloaded from ENA website as a TSV
11 ACCESSIONS_FILE='pputida_srr_ena_accessions.tsv'
12
13 # new directory 'reads_fastq' will be made to store all fastq files
14 java -jar ../ena-ftp-downloader/ena-file-downloader.jar
15 --accessions=$ACCESSIONS_FILE \\\
16 --format=$READS_FASTQ \\\
17 --location=$ROOT_PATH \\\
18 --protocol=FTP \\\
19 --asperaLocation=null \\\
20 --email=username@wpi.edu \\\
```

By default, a SLURM job will run for 24 hours at most. After 24 hours, the job will terminate. To prevent early termination of a job, we can request for more time by adding `#SBATCH -t 30:00:00`, for example. This will request 30 hours to run this job.

5. Quality Control with FastQC

Ideally, quality controls should be done at every analysis step. Listing 3 is an example script utilizing [FastQC](#) followed by [MultiQC](#) to aggregate multiple QC files into a single report.

```
1 ROOT_PATH='~/rnaseq'
2 FASTQ_RAW_DIR='reads_fastq'
3 FASTQ_TRIM_DIR='fastq_trim'
4 FASTQC_RESULTS='fastqc'
5 mkdir $FASTQC_RESULTS
6
7 # Run FastQC on all fastq.gz files in the specified directory
8 for filename in "$FASTQ_TRIM_DIR"/*.fastq.gz; do
9     # Run FASTQC on current file and extract to specified folder
10    fastqc "$filename" --extract -o "$(pwd)/$FASTQC_RESULTS"
11 done
12
13 # run MultiQC in the directory containing fastqc results
14 cd $FASTQC_RESULTS
15 multiqc .
```

6. Trimming Data and Removing Adapters

Trimmomatic is an NGS read preprocessing tool intended for trimming low quality Illumina data and removing adapter sequences [2]. Listing 4 is an example script that can distinguish between single-end and paired-end reads.

Software downloads and manual can be found on USADELLAB.org.

```
1 ROOT_PATH='~/rnaseq'
2 FASTQ_RAW_DIR='reads_fastq'
3
4 cd $FASTQ_RAW_DIR
5
6 # run Trimmomatic on all fastq.gz files in the directory
7 # first recursively go into subdirectories to find fastq files
8 find . -name "*.fastq.gz" | while read filename
9 do
10     echo $(basename ${filename%_*} ||)
11     # if paired-end reads, run the Trimmomatic command for PE
12     if [[ "$filename" == *_1.fastq.gz* ]];then
13         java -jar ~/Trimmomatic-0.39/trimmomatic-0.39.jar PE \\
14             $filename "$(basename ${filename%_*}_2.fastq.gz)" \\
15             "$(basename ${filename%_*}_paired_trimm1.fastq.gz)" \\
16             "$(basename ${filename%_*}_unpaired_trimm1.fastq.gz)" \\
17             "$(basename ${filename%_*}_paired_trimm2.fastq.gz)" \\
18             "$(basename ${filename%_*}_unpaired_trimm2.fastq.gz)" \\
19             ILLUMINACLIP:~/Trimmomatic-0.39/adapters/TruSeq3-PE.fa:
20             ↪ 2:30:10:2:True LEADING:3 TRAILING:3 MINLEN:36
21     # skipped the PE read ending with "_2.fastq.gz" because already
22     ↪ processed with the "_1.fq.gz"
23     elif [[ "$filename" == *_2.fastq.gz* ]];then
24         echo "Skipping $file ..."
25     # run Trimmomatic command for SE reads (file naming is a little messed
26     ↪ up, fix later)
27     else
28         java -jar ~/Trimmomatic-0.39/trimmomatic-0.39.jar SE -phred33 \\
29             $filename "`basename $filename`.trim.fastq.gz" \\
30             ILLUMINACLIP:~/Trimmomatic-0.39/adapters/TruSeq3-SE.fa:2:30:10
31             ↪ LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36
32     fi
33 done
34 cd $ROOT_PATH
35
36 # store all reads into new directory 'fastq_trim'
37 FASTQ_TRIM_DIR='fastq_trim'
38 mkdir $FASTQ_TRIM_DIR
```

```
36 cd $FASTQ_RAW_DIR
37 find . -maxdepth 1 -name "*.fastq.gz*" -exec mv -t
   ↪ $ROOT_PATH/$FASTQ_TRIM_DIR {} +
```

The script does the following:

1. Finds all files with any name ending in '.fastq.gz' in the directory specified with the variable FASTQ_RAW_DIR
2. For each file, it reads the file name. If a project contains paired-end reads, they are usually indicated as so in the file name (eg. 'sample_1.fastq.gz' for the forward template strand, and 'sample_2.fastq.gz' for the reverse)
3. If the read is paired-end, it will use the Trimmomatic tool specifically for paired-end reads. If it's single-end, it will utilize the single-end tool.
4. After trimming, all trimmed reads will be stored into its own directory, specified by FASTQ_TRIM_DIR

7. Mapping Reads with Alignment

To determine where on the *P. putida* Illumina reads are originating from, the tool STAR (Spliced Transcripts Alignment to a Reference) will be used to align reads to the reference genome [3].

In this case, BWA (Burrows-Wheeler Alignment) [4] is also a suitable alignment tool due to the short length of *P. putida*'s genome (*P. putida* is around 5.8 to 6.2 Mb, while the human genome is around 6.37 Gbp). However, if one would like to analyze larger genomes or discover non-canonical splices, STAR is the more preferred tool.

A [manual for STAR](#) can be found on Alexander Dobin's GitHub.

7.1 Download Genomic Data

To complete the alignment step, there are 2 additional files needed along with the Illumina reads: DNA sequence of your organism, and a gene annotations file. Both of these can be found on the [Pseudomonas Genome Database](#).

The DNA sequence needs to be a FASTA file (.fna). Below is a sample of this file:

```
1 >refseq|NC_002947.4|chromosome Pseudomonas putida KT2440 chromosome,
   ↪ complete genome. length=6181873;assembly=GCF_000007565.2
2 AACTGCTCCTCGGAAGTCGACCAACAAGTCAGCTATGACTTGGCATAATTTGTGCCGACA
3 AAATGCGCGCAGAGTATAGGGGTGGATTAACCCCTATTCAACTCTTCGGTAGTGATTTCC
4 GACTTCACGCTACAACAGGAATTGTTTTCAGCGGATGTGAGCGAGCACGCCTTGCAACTCG
5 TCAAGCGAGTTGTAGCGAATAACCAACTGGCCTTTGCCCTTGTTGCCATGACGGATCTGC
6 ACGGCCGAGCCCAGGGCTCTGCGAGCCGCTGTTCAAGGCGTGCGATATCCGGATCAGGT
7 TTGCTCGGTTTCGACCGGATCAGGCTTGTGCGTGAGCCACTGACGGACCAGTGCCCTCGGTT
```

```

8 TGGCGCACGGTGAGGCCACGTGCGACAACATGACGCGCCCCCTCCTCCTGACGATTTTCG
9 TCCAGGCCAGCAATGCACGGGCGTGCCCCATCTCCAGATCACCGTGGGCGAGCATGGTC
10 TTGATCGCATCGGGCAAGTGATGAGGCGCAGCAGTTGGCCACAGTCACCCGCGACTTG

```

“chromosome Pseudomonas putida KT2440 chromosome” specifies which chromosome the current sequence is located on. Since *P. putida* is a prokaryotic organism made up of a single chromosome, this will be the only instance we see this label. “assembly=GCF_000007565.2” refers to the unique NCBI RefSeq or GenBank assembly number assigned to the sequence. This number is dependent on the type of organism, type of sequence (genomic, RNA, or protein), etc. See [RefSeq Frequently Asked Questions \(FAQ\)](#) for more information.

chromosome	PseudomonasGenomeDB	CDS	147	1019	.	-	0
chromosome	PseudomonasGenomeDB	CDS	1029	1820	.	-	0
chromosome	PseudomonasGenomeDB	CDS	1839	2489	.	-	0
chromosome	PseudomonasGenomeDB	CDS	5012	6382	.	-	0
chromosome	PseudomonasGenomeDB	CDS	6471	8153	.	-	0
chromosome	PseudomonasGenomeDB	CDS	8156	8401	.	-	0
chromosome	PseudomonasGenomeDB	CDS	8812	8946	.	-	0
chromosome	PseudomonasGenomeDB	CDS	9542	11062	.	+	0
chromosome	PseudomonasGenomeDB	CDS	11103	12206	.	+	0
chromosome	PseudomonasGenomeDB	CDS	12222	13325	.	+	0

Fig. 5. First 10 Lines of GTF File - Columns 1-8

```

gene_id "PGD38821096"; transcript_id "PGD38821096"; locus_tag "PP_0001"; name "chromosome-partitioning protein"; replicon_xref "NC_002947.4";
gene_id "PGD38821098"; transcript_id "PGD38821098"; locus_tag "PP_0002"; name "chromosome partition protein"; replicon_xref "NC_002947.4";
gene_id "PGD38821100"; transcript_id "PGD38821100"; locus_tag "PP_0003"; name "16S RNA methyltransferase"; replicon_xref "NC_002947.4";
gene_id "PGD38821104"; transcript_id "PGD38821104"; locus_tag "PP_0005"; name "GTPase"; replicon_xref "NC_002947.4";
gene_id "PGD38821106"; transcript_id "PGD38821106"; locus_tag "PP_0006"; name "membrane protein insertase"; replicon_xref "NC_002947.4";
gene_id "PGD38821108"; transcript_id "PGD38821108"; locus_tag "PP_0007"; name "hypothetical protein"; replicon_xref "NC_002947.4";
gene_id "PGD38821112"; transcript_id "PGD38821112"; locus_tag "PP_0009"; name "50S ribosomal protein L34"; replicon_xref "NC_002947.4";
gene_id "PGD38821114"; transcript_id "PGD38821114"; locus_tag "PP_0010"; name "chromosomal replication initiator protein"; replicon_xref "NC_002947.4";
gene_id "PGD38821116"; transcript_id "PGD38821116"; locus_tag "PP_0011"; name "DNA polymerase III subunit beta"; replicon_xref "NC_002947.4";
gene_id "PGD38821118"; transcript_id "PGD38821118"; locus_tag "PP_0012"; name "DNA replication/repair protein"; replicon_xref "NC_002947.4";

```

Fig. 6. First 10 Lines of GTF File - Column 9

The genome annotations file needs to be a GTF (general transfer format). There are usually 9 columns in a GTF file (Fig. 5-6), with each column indicating the following:

- **seqname:** name of the chromosome or scaffold
- **source:** name of program that generated this feature of the data source
- **feature:** CDS, exon, ncRNA, tRNA, etc.
- **start:** Start position of the feature
- **end:** End position
- **score:** A floating point value. In this file, there is no score assigned to each feature, as indicated by ‘.’
- **strand:** Either + (forward) or - (reverse)

- **frame:** Labeled as '0', '1', or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on
- **attribute:** A semicolon-separated list of tag-value pairs, providing additional information about each feature.

Refer to E!Embl's page on [GFF/GTF File Format - Definition and supported options](#) for more information.

It is very important to make sure that the chromosome names in the column of the GTF annotation file matches the chromosome names in the FASTA genome sequence file (the string after "}" and before the first space). Even though *P. putida* contains only one chromosome, the indexing step will not be successful if these names don't match. In this case there is a mismatch between our FASTA and GTF files, and we will need to change 'refseq—NC_002947.4—chromosome' on our FASTA file to 'chromosome'. To make change, we can use `sed -e`:

```
1 sed {e `s/original\_string/desired\_string/' genome_file.fna
```

7.2 STAR Indexing

The following script takes the FASTA file with the DNA sequence and the GTF file with genomic annotations and the basic minimum options on STAR to generate genome indices. To read more in depth about these options and other advanced options (eg. which chromosomes to include, annotations in GFF format, very small genome, etc.), refer to the [STAR manual](#).

```
1 #!/bin/bash
2 #SBATCH -N 2
3 #SBATCH -n 4
4 #SBATCH --mem=0
5
6 FASTA='~/Pseudomonas_putida_KT2440_110.fna'
7 GTF='~/Pseudomonas_putida_KT2440_110.gtf'
8 # directory where STAR index will be placed
9 INDEX_DIR='~/STARindex'
10
11 # run STAR in "genomeGenerate" mode
12 STAR --runMode genomeGenerate \\\
13 --runThreadN 1 \\\
14 --genomeDir ${INDEX_DIR} \\\
15 --genomeFastaFiles ${FASTA} \\\
16 --sjdbGTFfile ${GTF} \\\
17 --sjdbOverhang 49 \\\
18 --quantMode GeneCounts \\\
19 --genomeSAindexNbases 10
```

7.3 STAR Alignment

There are 3 basic options required to run a mapping job:

```
1 --runThreadN NumberOfThreads
2 --genomeDir /path/to/genomeDir
3 --readFilesIn /path/to/read1 [/path/to/read2]
```

For running mapping jobs on samples with different strains, this code assumes that you have the TSV file with each sample's accession number and project description. It will locate the accession number in the FASTQ file name, match it with the description on the TSV. From there it will try to identify strain names and run the mapping job with the best matched strain.

```
1 ROOT_PATH='~/rnase2'
2 FASTQ_RAW_DIR='reads_fastq'
3 FASTQ_TRIM_DIR='fastq_trim'
4 FASTQC_RESULTS='fastqc'
5
6 index_KT2440='~/strain_KT2440/index_STAR'
7 index_BIRD1='~/strain_BIRD1/index_STAR'
8 index_S12='~/strain_S12/index_STAR'
9
10 ALIGN_DIR='align_star'
11 mkdir -p "$ROOT_PATH/$ALIGN_DIR"
12 cd "$ROOT_PATH/$ALIGN_DIR"
13
14 for SAMPLE in "$ROOT_PATH/$FASTQ_TRIM_DIR"/*.fastq.gz
15 do
16     base_file_name=$(basename ${SAMPLE%.*})
17     project_name=\${base_file_name%_*}
18     DESC=$(awk -v x="\${project_name}" -F '\t' '$1~x {print $2}'
19         ↪ ${ROOT_PATH}/pputida_srr_ena_accessions.tsv)
20
21     # Check if output BAM file already exists (ie. check is sample is
22     ↪ already mapped)
23     if [ ! -e "${ROOT_PATH}/${ALIGN_DIR}/
24         ↪ ${base_file_name}_Aligned.sortedByCoord.out.bam" ]; then
25         echo "Calculating alignment for $SAMPLE..."
26
27         # Run the alignment with appropriate strain based on project
28         ↪ description
29         if [[ "$DESC" == *"KT2440"* ]]; then
30             echo "KT2440"
```

```

27     STAR --genomeDir $index_KT2440 --readFilesIn $SAMPLE
      ↪ --readFilesCommand zcat --outFileNamePrefix
      ↪ ${base_file_name}_ --outFilterMultimapNmax 1
      ↪ --outReadsUnmapped Fastx --outSAMtype BAM
      ↪ SortedByCoordinate --twopassMode Basic --runThreadN 2
      ↪ --limitBAMsortRAM 1700000000
28     elif [[ "$DESCR" == *"BIRD"* ]]; then
29         echo "BIRD-1"
30     STAR --genomeDir $index_BIRD1 --readFilesIn $SAMPLE
      ↪ --readFilesCommand zcat --outFileNamePrefix
      ↪ ${base_file_name}_ --outFilterMultimapNmax 1
      ↪ --outReadsUnmapped Fastx --outSAMtype BAM
      ↪ SortedByCoordinate --twopassMode Basic --runThreadN 2
      ↪ --limitBAMsortRAM 1700000000
31     elif [[ "$DESCR" == *"S12"* ]]; then
32         echo "S12"
33     STAR --genomeDir $index_S12 --readFilesIn $SAMPLE
      ↪ --readFilesCommand zcat --outFileNamePrefix
      ↪ ${base_file_name}_ --outFilterMultimapNmax 1
      ↪ --outReadsUnmapped Fastx --outSAMtype BAM
      ↪ SortedByCoordinate --twopassMode Basic --runThreadN 2
      ↪ --limitBAMsortRAM 1700000000
34     fi
35     else
36         echo "Alignment already exists for $SAMPLE. Skipping..."
37     fi
38 done

```

To generate high quality mapped reads, use Samtools [5]:

```

1     ROOT_PATH='./rnaseq'
2     FASTQ_RAW_DIR='reads_fastq'
3     FASTQ_TRIM_DIR='fastq_trim'
4     FASTQC_RESULTS='fastqc'
5     ALIGN_DIR='align_star'
6     ALIGN_HQ_DIR='indexed_HQ'
7     mkdir $ALIGN_HQ_DIR
8     cd $ALIGN_HQ_DIR
9
10    for SAMPLE in "$ROOT_PATH"/"$ALIGN_DIR"/*Aligned.sortedByCoord.out.bam
11    do
12        # shave down filename to just its SRR # (eg.
      ↪ 'SRR6012666_2.fastq.gz' to just 'SRR6012666')
13        base_file_name=$(basename ${SAMPLE%.*})
14        project_name=\${base_file_name%_*}
15        echo "${project_name}"

```

```
16 samtools view -h -b -q 20 $SAMPLE >
    ↪ \${base_file_name}_high_mapq_reads.bam
17 done
```

The code uses the following options from Samtools:

1. view Views and converts SAM/BAM/CRAM files
2. -h, --with-header Include the header in the output
3. -b, --bam Output in BAM format
4. -q INT, --min-MQ INT Skips alignments with MAPQ smaller than INT (the specified integer)

Refer to the [Samtools documentation](#) for more details and other options

8. Generate Read Count Table

This is the final step within the command line! After this, we will finally have our table of raw read counts for each gene in rows and each condition in columns. We will do this using the `featureCounts` software, which counts RNA-seq reads to each gene on the genome [6]. Refer to the [Rsubread/Subread Users Guide](#) for more information.

```
1 gtf_KT2440=' ../KT2440_110.gtf'
2 READ_COUNTS_DIR='read_counts'
3 mkdir $READ_COUNTS_DIR
4 cd $READ_COUNTS_DIR
5
6 featureCounts -a $gtf_KT2440 -o ./featureCounts_results_*.txt
    ↪ \${ROOT_PATH}/\${ALIGN_HQ_DIR}/*bam -t CDS
```

Below is a sample output after running `featureCounts`, where the first line is the name of each column:

Geneid	Chr	Start	End	Strand	Length	read1.bam	...
PGD38821096	chromosome	147	1019	-	873	188	...
PGD38821098	chromosome	1029	1820	-	792	157	...
PGD38821100	chromosome	1839	2489	-	651	141	...
PGD38821104	chromosome	5012	6382	-	1371	129	...
PGD38821106	chromosome	6471	8153	-	1683	377	...
PGD38821108	chromosome	8156	8401	-	246	73	...
PGD38821112	chromosome	8812	8946	-	135	362	..
PGD38821114	chromosome	9542	11062	+	1521	611	..
PGD38821116	chromosome	11103	12206	+	1104	280	..

9. Differential Gene Expression Analysis (DGE)

There are a variety of analyses one can do with RNA-seq, but one of the most popular ones count data is the identification of differentially expressed genes, or genes that have significant upregulation or downregulation when comparing two conditions with each other.

R [7], a language and environment for statistical computing, is one of the most favored languages for Bioinformatics and has the most dedicated and supported tools for such analyses. If you are new to programming and would like to better understand your dataset, R is a good place to start.

As of now, there are three tools that perform the best for DGE analysis: edgeR [8], DESeq2 [9], and limma-voom [10]. They all implement several statistical methods that have been developed and refined over two decades. Each has their own user manual and user guide to reference for analyses, including [Michael Love's detailed vignette on analyzing RNA-seq data with DESeq2](#).

References

- [1] Stark, R., Grzelak, M., & Hadfield, J. (2019). Rna sequencing: the teenage years. *Nature Reviews Genetics*, 20(11), 631–656.
- [2] Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, 30(15), 2114–2120.
- [3] Dobin, A., & Gingeras, T. R. (2015). Mapping rna-seq reads with star. *Current protocols in bioinformatics*, 51(1), 11–14.
- [4] Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *bioinformatics*, 25(14), 1754–1760.
- [5] Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., et al. (2021). Twelve years of samtools and bcftools. *Gigascience*, 10(2), giab008.
- [6] 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.
- [7] R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. <https://www.R-project.org/>
- [8] Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *bioinformatics*, 26(1), 139–140.
- [9] Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15, 1–21.
- [10] Law, C. W., Chen, Y., Shi, W., & Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for rna-seq read counts. *Genome biology*, 15, 1–17.