Introducing the Aizawa et al., (2022) dataset

In the study by Aizawa and colleagues, the authors showed that they were able to establish patient-derived xenograft (PDX), patient-derived organoids (PDO) and PDX-derived organoids (PDXO) as experimental models of human salivary gland cancer. Their bioinformatics pipeline involved classifying human (graft) from mouse (host) reads for all 13 paired-end RNA-seq samples (including the PDO samples which consist purely of human reads).

In this mini-project/tutorial, I repeat the authors' analysis pipeline but utilise the recently published `Xengsort` tool (Zentgraft and Rahman, 2021) for xenograft sorting (instead of `Xenome` which was used by the authors), and compare my results with the publication. I have included the bash scripts in this document for convenience, but all the scripts and nohup output files can be located in my github repository.

Downloading data with SRA explorer

Under the Data Availability section of the manuscript, Aizawa and colleagues stated that the bulk RNA sequencing data for the study was deposited in the "DNA Data Bank of Japan Sequence Read Archive (DRA) under accession number `DRA011243`".

Hence, I entered this into the excellent SRA-explorer tool by Phil Ewels, which conveniently identified the links to the raw fastq files. I then added the 13 datasets to collection (Figure 1) and ran the automatically generated bash script (Figure 2) which downloads the fastq files with curl.

SRA Explorer

This tool aims to make datasets within the Sequence Read Archive more accessible.

Search for: DRA011243[All Fields]

Max Results 100 **Start At Record** 0

Need inspiration? Try `GSE30567`, `SRP043510`, `PRJEB8073`, `ERP009109` or `human liver miRNA`.

Select relevant datasets and click *add to collection*. When you're finished, view all saved datasets with the button in the top right of the page, where you can copy the SRA URLs.

Showing 13 results.

Filter results: Enter search term | All Fields | Add 0 to collection

Title	Accession	Instrument	Total Bases (Mb)	Date Created
Illumina NovaSeq 6000 paired end sequencing of SAMD00263607	DRR259325	Illumina NovaSeq 6000	110518	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263608	DRR259326	Illumina NovaSeq 6000	102812	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263609	DRR259327	Illumina NovaSeq 6000	114829	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263610	DRR259328	Illumina NovaSeq 6000	130824	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263611	DRR259329	Illumina NovaSeq 6000	79311	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263612	DRR259330	Illumina NovaSeq 6000	69413	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263613	DRR259331	Illumina NovaSeq 6000	124383	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263614	DRR259332	Illumina NovaSeq 6000	115310	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263615	DRR259333	Illumina NovaSeq 6000	124843	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263616	DRR259334	Illumina NovaSeq 6000	100189	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263617	DRR259335	Illumina NovaSeq 6000	106978	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263618	DRR259336	Illumina NovaSeq 6000	102365	29 Jul 2021
Illumina NovaSeq 6000 paired end sequencing of SAMD00263619	DRR259337	Illumina NovaSeq 6000	105688	29 Jul 2021

Figure 1: Screenshot of the SRAexplorer page, showing the 13 datasets selected for download

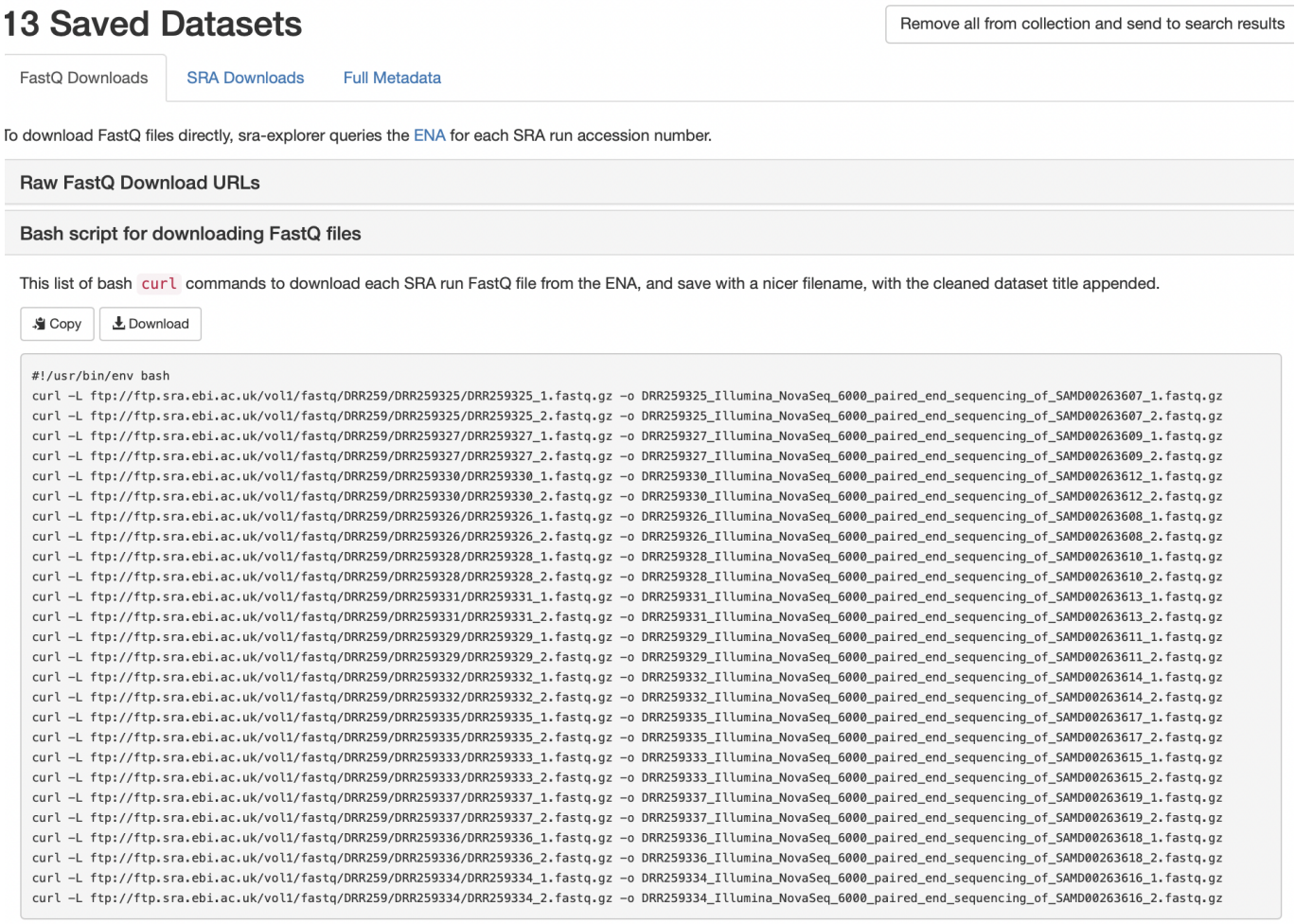

Figure 2: Screenshot of the SRAexplorer page, showing the auto-generated bash script for fastq file download

Running the following command:

```
du -ch *{1,2}.fastq.gz
```

in the directory shows that the total size of the 13x2 fastq files is about 60G.

QC-ing with fastp

Similar to the paper, I performed basic QC with `fastp`.

01.fastp.sh

```
#!/bin/bash

DIR=/media/gedac/kane/raw_data/xenograft_DRA011243

for fq1 in $DIR/*_1.fastq.gz; do
    fq2=${fq1%_1.fastq.gz}_2.fastq.gz
    fastp -i $fq1 -I $fq2 -o ${fq1%.fastq.gz}.trimmed.fastq.gz -O
```

```
${fq2%.fastq.gz}.trimmed.fastq.gz
done
```

Installing and running Xengsort

I searched for bioinformatics tools that can assist in distinguishing between human and mouse-derived reads (xenograft sorting). There are two general approaches to this (summarised in Zentgraft and Rahman, 2021): either alignment-based or alignment-free. Alignment-based methods take as input the aligned BAM files and assesses whether each mapped read maps better to the host or graft genome. In contrast, alignment-free methods take the raw fastq files as input, creates k-mers of the raw reads, references a large lookup table to assign species information to the k-mer and finally sorts reads into different categories based on its k-mer composition. I was keen to try out the alignment-free methods because it was novel to me.

In particular, I decided to use `Xengsort` over `Xenome` despite the higher number of citations for the latter tool for several reasons. First and foremost, `Xenome` was not easy to install at all. There was no gitlab/github repostiory and no way to download the software via conda / pip. Upon digging deeper, installing it also requires installing the entire Gossamer bioinformatics suite bundle. In contrast, `Xengsort` had a gitlab repository with clear instructions, as well as a conda *requirements.txt* file. In addition, the `Xengsort` publication was released in 2021 with very recent gitlab commits, whereas `Xenome` was published in 2012. `Xengsort` it is!

Acquiring the reference fasta files from GENCODE

A first attempt

To align reads to the human and mouse genomes, I downloaded the reference fasta files, specifically the genome primary assembly (no patches or haplotypes), from Gencode. I used Release 42 (GRCh38.p13) and Release M31 (GRCm39) which are the latest versions at the time of writing.

First, I generated the indices with `Xengsort index` and then run `Xengsort classify` on each of the fastq file pairs. However, the results indicate that a majority of the reads were classified as belonging to neither human/mouse samples, which is questionable as I expected a greater proportion of reads to be assigned to the human/graft category. Indeed, from supplementary fig S4a-b, I confirmed that this should be the case as this was the authors' result when they analysed their files with `xenome`. This meant that something went wrong with my analysis, and upon reflection, I had a suspicion that this was due to the `xengsort index` step.

A second attempt

I then checked the `gitlab` page more thoroughly and discovered this issue, in which the poster uses the `top-level` and `cdna` reference sequences during the indexing step. Indeed, this code was motivated from the authors' snakefile.

Here is a script used to download the appropriate reference files:

```
# Top-level fasta file for GRCh38
nohup curl https://ftp.ensembl.org/pub/release-
108/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.toplevel.fa.gz -o
Homo_sapiens.GRCh38.dna.toplevel.fa.gz &> nohup.grch38toplevel.out &

# cDNA fasta file for GRCh38
nohup curl https://ftp.ensembl.org/pub/release-
108/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh38.cdna.all.fa.gz -o
Homo_sapiens.GRCh38.cdna.all.fa.gz &> nohup.grch38cdna.out &

# Top-level fasta file for GRCm39
nohup curl https://ftp.ensembl.org/pub/release-
108/fasta/mus_musculus/dna/Mus_musculus.GRCm39.dna.toplevel.fa.gz -o
Mus_musculus.GRCm39.dna.toplevel.fa.gz &> nohup.grcm39toplevel.out &

# cDNA fasta file for GRCm39
nohup curl https://ftp.ensembl.org/pub/release-
108/fasta/mus_musculus/cdna/Mus_musculus.GRCm39.cdna.all.fa.gz -o
Mus_musculus.GRCm39.cdna.all.fa.gz &> nohup.grcm39cdna.out &
```

Of course, the next question to ask is why the genome reference files alone would not suffice. As always, someone else had a similar question and the authors responded here.

> I would say, the transcriptome is necessary for RNA-seq, as **some exons are quite short, and you may miss many k-mers that only occur in the transcripts (across introns)**, but never in the genomic DNA. Note that **genomic DNA is still useful as some of the RNAs may not have been completely spliced yet**. - Sven Rahmann, author of XEngsort.

After rerunning the `xengsort index` and `xengsort classify` steps, I got a much higher number of reads that are assigned to the `graft` category. The final scripts used to run the `xengsort` steps are:

02.xengsort.index.sh

```
#!/bin/bash

nohup xengsort index \
--index human_mouse_index.zarr \
-G <(zcat Homo_sapiens.GRCh38.cdna.all.fa.gz
Homo_sapiens.GRCh38.dna.toplevel.fa.gz) \
-H <(zcat Mus_musculus.GRCm39.cdna.all.fa.gz
Mus_musculus.GRCm39.dna.toplevel.fa.gz) \
-n 4496607845 \
-p 4 \
--fill 0.88 \
-W 11 \
&> nohup.xengsort.index.out &
```

03.xengsort_classify_script.sh

```
#!/bin/bash
DIR=/media/gedac/kane/raw_data/xenograft_DRA011243

for fq1 in $DIR/*_1.trimmed.fastq.gz; do
    fq2=${fq1%_1.trimmed.fastq.gz}_2.trimmed.fastq.gz
    xengsort classify --index human_mouse_index.zarr --fastq <(zcat $fq1)
--pairs <(zcat $fq2) --out ${fq1%_1.trimmed.fastq.gz} -T 8
done
```

Running STAR alignment on human (graft) reads

After `Xengsort classify` separated the reads into the 5 categories (`host`, `graft`, `both`, `neither` and `ambigiuous`), I then aligned the human (graft) reads to the human reference genome using `STAR`. But before the alignment, I have to first generate the STAR genome index.

04.star.genomegenerate.sh

```
#!/bin/bash

STAR --runThreadN 12 \
    --runMode genomeGenerate \
    --genomeDir star_index_hg38 \
    --genomeFastaFiles grch38.primary_assembly.fa \
    --sjdbGTFfile gencode.v42.primary_assembly.annotation.gtf
```

05.star.align.default.sh

```
#!/bin/bash
DIR=/media/gedac/kane/raw_data/xenograft_DRA011243

for fq1 in $DIR/*-graft.1.fq; do
    fq2=${fq1%-graft.1.fq}-graft.2.fq
    STAR --genomeDir
/media/gedac/kane/raw_data/xenograft_DRA011243/star_index_hg38 \
    --runThreadN 20 \
    --readFilesIn $fq1 $fq2 \
    --outSAMtype BAM Unsorted \
    --outFileNamePrefix ${fq1%-graft.1.fq}-default_graft
done
```

Examining the `Log.final.out` files, I was satisfied with the % of uniquely-mapped reads across the samples with the default parameter settings for STAR. Indeed, the authors also did not mention that they tweaked the parameters from default. Out of curiosity, I tried reducing the `--outFilterScoreMinOverLread` and `--outFilterMatchNminOverLread` parameters from the default value of 0.66 to 0.50 in order to reduce the % of unmapped reads : too short. This reduces the strigency of the filter as only 50% of the total read length (approx 150 bp in our example) needs to be mapped Biostars

issue. Nevertheless, I stuck to the default settings as these parameter changes did not lead to a notable change in the results.

Inferring strandedness with infer_experiment.py

Under the materials and methods section, I identified that the RNAseq libraries were constructed using the Illumina TruSeq Stranded mRNA kit. Stranded RNA-seq libraries preserve the directionality of the transcripts, which allows for the identification of the strand of origin for each transcript, and can thus distinguish between sense and antisense transcripts. A quick search reveals that the TruSeq Stranded mRNA kit is reverse-stranded, and that we should specify this in featureCounts with the `-s 2` flag. I quickly verified this by running the `infer_experiment.py` script from `RSeQC` on the `DRR259325` bam file, which shows that around 86% of the paired-end reads were identified as `1+-,1--,2++,2--`, confirming that the library is reverse-stranded since read 2 aligns with the RNA strand (Figure 3).

```
This is PairEnd Data
Fraction of reads failed to determine: 0.1300
Fraction of reads explained by "1++,1--,2+-,2-+": 0.0098
Fraction of reads explained by "1+-,1-+,2++,2--": 0.8602
```

Figure 3: Output of infer_experiment.py from RSeQC when ran on the DRR259325 aligned BAM file

06.infer.strandedness.sh

```
#!/bin/bash

# Generate the reference bed file from the GENCODE GTF file
gtf2bed < gencode.v42.primary_assembly.annotation.gtf >
gencode.v42.primary_assembly.annotation.bed

# Check the strandedness from a randomly chosen .bam file
infer_experiment.py -r gencode.v42.primary_assembly.annotation.bed -i
DRR259325_Illumina_NovaSeq_6000_paired_end_sequencing_of_SAMD00263607Align
ed.out.bam
```

Quantifying reads with featureCounts

07.featurecounts.sh

```
#!/bin/bash

DIR=/media/gedac/kane/raw_data/xenograft_DRA011243

featureCounts --verbose \
-T 12 \
-t exon \
--extraAttributes gene_name \
-g gene_id \
-s 2 \
```

```
-F GTF \
-p --countReadPairs \
-a gencode.v42.primary_assembly.annotation.gtf \
-o counts.txt \
$(find . -name "*default_graft*.bam") 2> run.log
```

Results comparison

Xenograft sorting

To compare the results of `Xengsort` vs `Xenome` , I plotted a similar plot to supplementary figure 4b (shown in Figure 4 here) which examines the proportion of reads that were classified by the sorting tool into 5 different categories.

The results for both tools are similar. For example, the patient with label `YCU-ACC-4` has around 30.4 % of reads classified as mouse (host) (Figure 4), which is a substantially larger percentage than the other samples. This idiosyncracy is reflected in my analysis with `Xengsort`, where 31.2 % of reads from `YCU-ACC-4` are assigned to the host (mouse) (Figure 5D).

In fact, `xengsort` appears to have a greater accuracy than `Xenome`. As patient-derived organoids (PDO) consist purely of human tissue samples, we know that reads originating from PDO samples that are not classified as human (graft) reads are incorrectly classified. Across each of the 4 PDO samples, `Xengsort` classifies around 99.2 - 99.5 % of reads as human (Figure 5D, PDO panel) whereas `Xenome` classifies a lower percentage (96.9 - 97.3 %) of reads as human (Figure 4). Hence, `xengsort` appears to have a better accuracy over `xenome`.

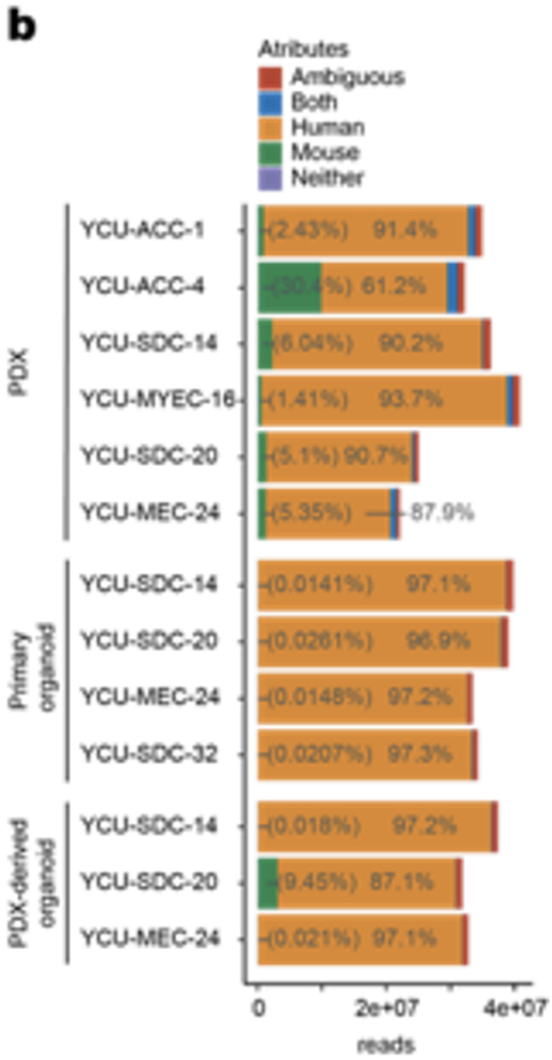

Figure 4: Supplementary Figure S4b (Aizawa et al., 2022)

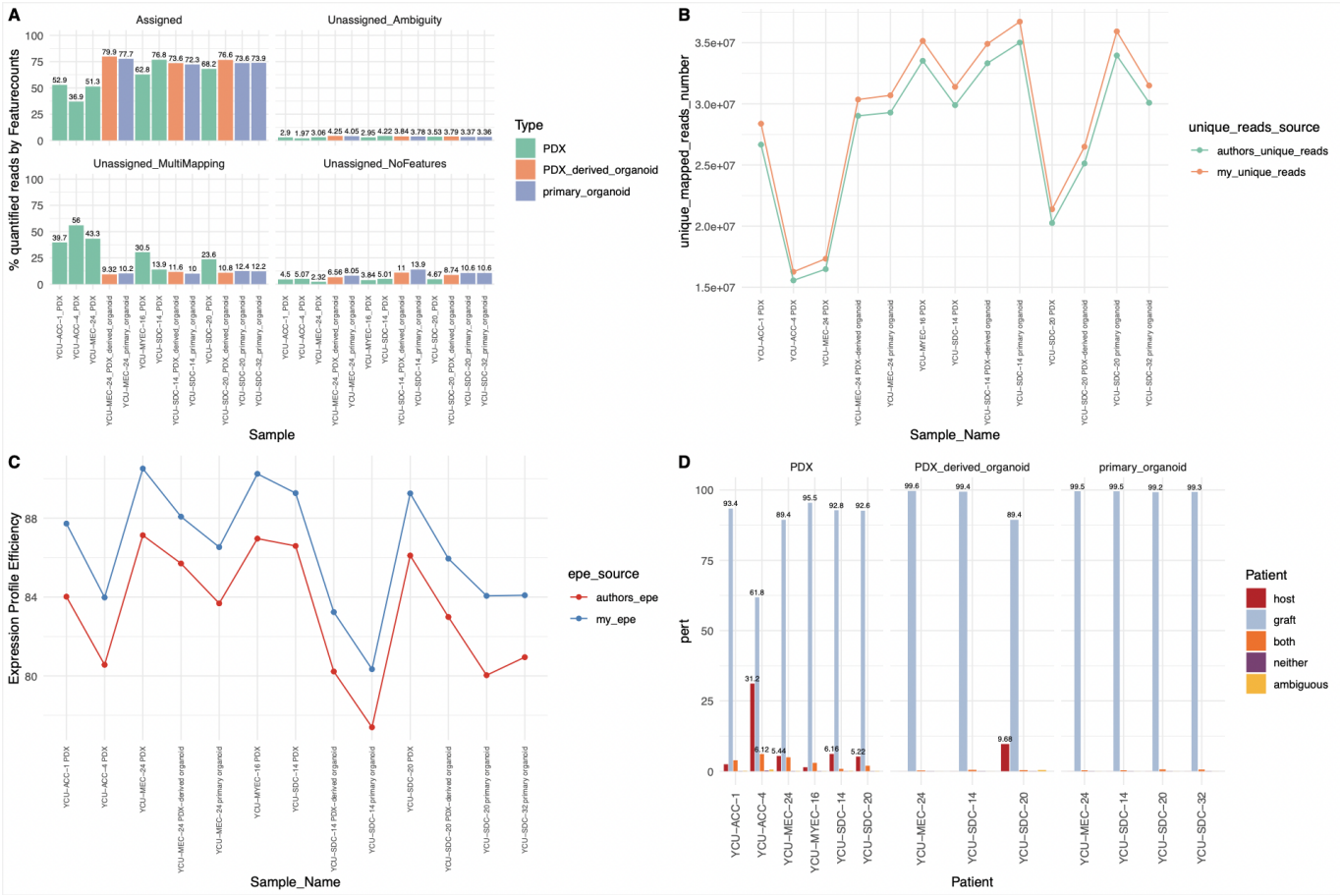

Figure 5: Read mapping and classification statistics. A) Summary of FeatureCounts read summarisation B) Comparison of the number of uniquely mapped reads C) Comparison of the expression profile efficiency scores D) Results of Xengsort classification. Only percentage scores >= 5% are annotated on the bar plot.

Read mapping and quantification results

I also compared the number of uniquely mapped reads by STAR, as well as the ratio of exon-mapped reads to total uniquely mapped reads (expression profile efficiency), with the authors' results as reported in supplementary table S3. Both the number of uniquely mapped reads (Figure 5B) and the expression profile efficiency (Figure 5C) in my analysis pipeline was higher than the authors' analysis. I also share the `featureCounts` read summarisation results to display the distribution of reads across the assigned and unassigned categories (Figure 5A). As the STAR alignment was carried out after xenograft sorting with `Xengsort`, I believe that this improvement over the authors' reported results could be attributed to better read classification by `Xengsort` over `Xenome`.

Code availability

All the code used to conduct the analysis can be found in this github repository link.

References

- Establishment of experimental salivary gland cancer models using organoid culture and patient-derived xenografting : Aizawa et al, (2022)
- featureCounts: an efficient general purpose program for assigning sequence reads to genomic features : Liao et al., (2014)

- Fast lightweight accurate xenograft sorting : Zentgraf and Rahman (2021)
- STAR: ultrafast universal RNA-seq aligner: Dobin et al., (2013)
- How does the TruSeq stranded mRNA sequencing kit work?