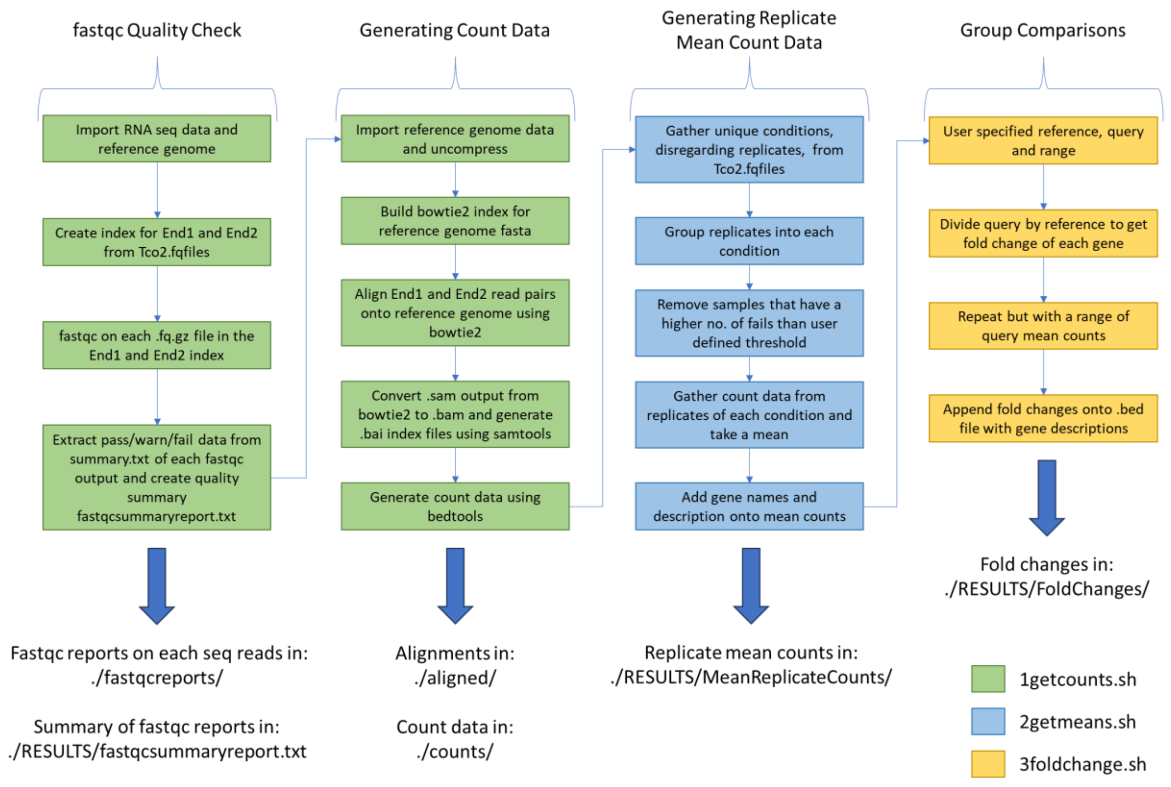
# foldchange.sh

*Figure 1: Flowchart of Pipeline to Generate Alignments, Counts, and Fold Changes of data from an RNA-seq Experiment*



fastqc Quality Check
Generating Count Data
Generating Replicate Mean Count Data
Group Comparisons
Import RNA seq data and reference genome
Create index for End1 and End2 from Tco2.fqfiles
fastqc on each .fq.gz file in the End1 and End2 index
Extract pass/warn/fail data from summary.txt of each fastqc output and create quality summary fastqcsummaryreport.txt
Import reference genome data and uncompress
Build bowtie2 index for reference genome fasta
Align End1 and End2 read pairs onto reference genome using bowtie2
Convert .sam output from bowtie2 to .bam and generate .bai index files using samtools
Generate count data using bedtools
Gather unique conditions, disregarding replicates, from Tco2.fqfiles
Group replicates into each condition
Remove samples that have a higher no. of fails than user defined threshold
Gather count data from replicates of each condition and take a mean
Add gene names and description onto mean counts
User specified reference, query and range
Divide query by reference to get fold change of each gene
Repeat but with a range of query mean counts
Append fold changes onto .bed file with gene descriptions
Fastqc reports on each seq reads in: ./fastqcreports/
Summary of fastqc reports in: ./RESULTS/fastqcsummaryreport.txt
Alignments in: ./aligned/
Count data in: ./counts/
Replicate mean counts in: ./RESULTS/MeanReplicateCounts/
Fold changes in: ./RESULTS/FoldChanges/
🟩 1getcounts.sh
🟦 2getmeans.sh
🟨 3foldchange.sh

*Figure 1 Legend: High level overview of pipeline. Big blue arrows indicate outputs of interest and their respective file locations. Each process color coded to show which script file it is run in.*

## **<u>GOI_crawler.R</u>**

<u>Script Overview</u>

*Figure 2: Flowchart of data collection script*

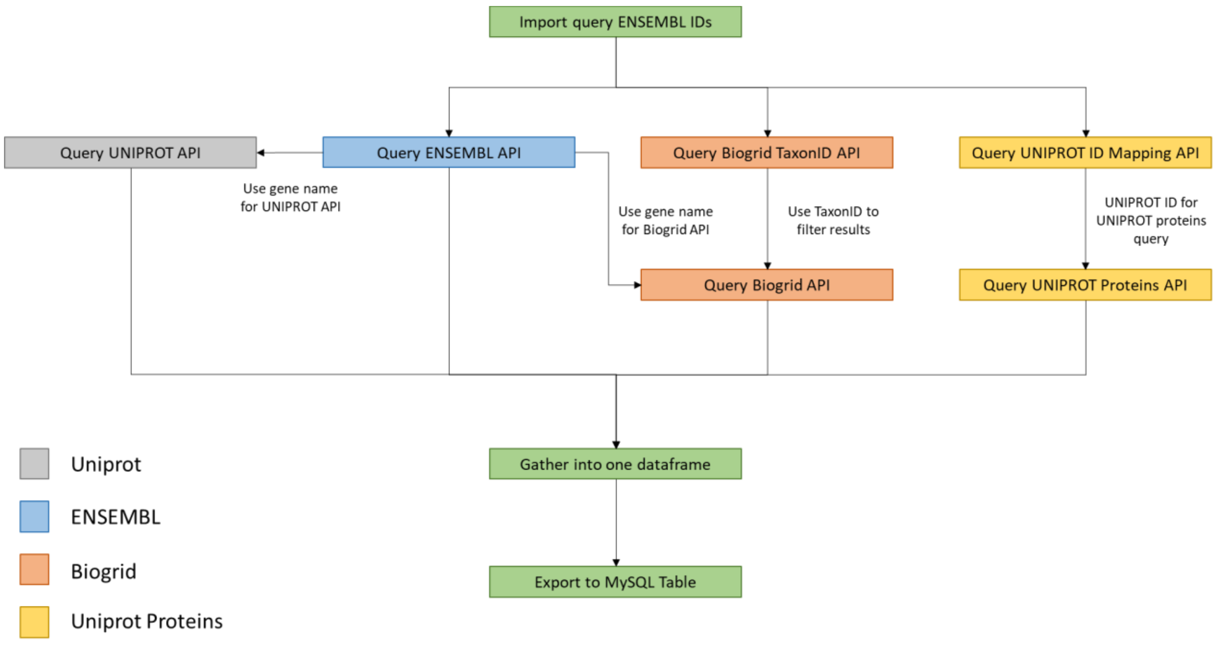

Import query ENSEMBL IDs
Query UNIPROT API
Query ENSEMBL API
Query Biogrid TaxonID API
Query UNIPROT ID Mapping API
Use gene name for UNIPROT API
Use gene name for Biogrid API
Use TaxonID to filter results
UNIPROT ID for UNIPROT proteins query
Query Biogrid API
Query UNIPROT Proteins API
Gather into one dataframe
Export to MySQL Table
Uniprot
ENSEMBL
Biogrid
Uniprot Proteins

*Figure 2 Legend: Flowchart of program showcasing the databases queried, different API access points, and how information flows between different processes. The color of the boxes (except green) indicates the database being queried. Script takes query Ensembl IDs and queries it against the Ensembl API. It also queries the Biogrid taxonID API to get the taxonID to query against the Biogrid API along with the gene name from the Ensembl API so it selects only organism specific information. The gene name from the Ensembl API is also used to query against the UNIPROT API to extract data. The Ensembl ID is also used to query against the UNIPROT ID mapping API to get a uniprot ID to query against the UNIPROT Proteins API. All relevant data is extracted and put into a data frame where it is exported to a MySQL database.*

*Figure 3: Final MySQL table columns and their descriptions*

| **Column Name** | **Description** |
|---|---|
| EnsemblID | Ensemble ID (From: Imported query genes) |
| UniprotID | Uniprot ID (From: Uniprot ID mapping API) |
| GeneName | Gene common name (From: Ensembl API) |
| GeneProduct | Product from gene (From: Ensembl API) |
| ProteinName | Protein Name (From: Uniprot Proteins) |
| UniparcID | Uniparc ID (From: Uniprot API) |
| RefseqID | Refseq ID (From: Uniprot API) |
| pdbID | Protein Data Base ID (From: Uniprot API) |
| BiogridID | Biogrid ID (From: Uniprot API) |
| StringID | String ID (From: Uniprot API) |
| goID | Gene ontology ID (From: Uniprot API) |

| | |
|---|---|
| goDescription | Gene ontologies with their descriptions |
| EnsemblObjectType | Ensembl ID Object (From: Ensembl API) |
| GeneDescription | Description of gene (From: Ensembl API) |
| Species | Origin species of query gene (From: Ensembl API) |
| TaxonID | Taxonomy ID of query origin species (From: Biogrid TaxonID API) |
| DNASeq | DNA Sequence (bases) (From: Ensembl API) |
| ChromosomeLoci | Chromosome gene is located in (From: Ensembl API) |
| Strand | Strand (From: Ensembl API) |
| StartPos | Start position of gene (From: Ensembl API) |
| EndPos | End position of gene (From: Ensembl API) |
| Keywords | Keywords associated with protein (From: Uniprot Proteins) |
| ProteinFamily | Protein family (From: Uniprot API) |
| ProteinSeq | Protein Sequence (From: Uniprot Proteins) |
| ProteinLength | Protein length (residues) (From: Uniprot API) |
| ProteinMass | Protein mass (Da) (From: Uniprot API) |
| ProteinFunction | Protein function description (From: Uniprot API) |
| ProteinFeatures | Protein features description (From: Uniprot API) |
| ProteinSubunits | Protein subunit information (From: Uniprot API) |
| ProteinDevStage | Developmental stage protein is associated with (From: Uniprot API) |
| ProteinTissueLoci | Tissue specificity of protein (From: Uniprot API) |
| ProteinSubCellLoci | Sub cellular location of protein (From: Uniprot API) |
| ProteinResidueMod | Protein residue modifications (From: Uniprot API) |
| ProteinPTM | Post translational modification of protein (From: Uniprot API) |
| Interactions | Proteins which interact with query (From: Biogrid API) |

Important Design Features

1. Queries 4 databases, using 7 API access points to gather data.
    a. Databases used: Ensembl (Martin et al., 2023), Biogrid (Oughtred et al., 2021), Uniprot (The UniProt Consortium, 2023), Uniprot Proteins (Containing Uniprot sequences, + other services imported from Large Scale Data Sources (LSS) including: 1000Genomes, ExAC, PeptideAtlas etc.) (Nightingale et al., 2017).
    b. API Access Points used: Ensembl Lookup REST API, Ensembl Sequence REST API, Uniprot website REST API, Uniprot Proteins REST API, Biogrid organisms REST API, Biogrid REST API.
2. Packages used: httr (Wickham, 2023), jsonlite (Ooms, 2014), RCurl (Lang, 2023), curl (Ooms, 2023), RMySQL (Ooms et al., 2023), queryup (Voisinne, 2019)
3. Gene queries can be set by the user, with either modifying the querygenes variable or by importing a .csv file with all the Ensembl IDs for the queries. Query genes can be set by the user with different query genes and different number of query genes (Number of query genes can be scalable from a small amount to a very large amount of Ensembl IDs, only limited by API limits). No information about the queries are hard coded, meaning all the information is automatically extracted from the Ensembl IDs only.
4. Not all genes are protein coding, and since the information extracted focuses on proteins, it is able to still process genes that are not protein coding. (eg. snoRNAs, pseudogenes etc.). Also able to process returns from APIs that are different eg. For empty or variable biogrid returns (eg. Different number of interacting proteins between queries), failed ID mapping, different uniprot protein returns etc.

5. Showcases 2 different ways of connecting to an API: url generation and packages such as queryup.
6. Showcases GET and POST requests for APIs.
7. Progress is printed onto terminal to show user the progress of the script.
8. Extracts information from Uniprot only on the main protein, and discards information about various isoforms relating to a gene name.
9. Sys.sleep() to avoid rate limitation from APIs. Time is set via sleeptime variable and is default to 0.1
    a. Ensembl: 55000 requests per hr (Yates et al., 2015) therefore minimum sleeptime is Sys.sleep(0.0655)
    b. Assumed other APIs followed similar rate limits therefore set default to 0.1s

## Hypothetical Use Case

Biological question:

A differential expression RNA seq experiment has identified several genes, along with their ensembl IDs, that are differentially expressed after a change in condition (eg. Disease, treatment, time etc.). These genes may produce proteins which help the organism to survive better, and researchers want to know more about how the gene helps the organism survive. They may also want to use the gene products as drug targets or target genes for synthetic biology. Researchers want to find out more about the identified genes to find:

1. Gene and product names, information and descriptions
    a. GeneName, GeneProduct, ProteinName, EnsemblObjectType, GeneDescription, Species, ChromosomeLoci
2. Gene and gene product IDs to signpost further information in other databases
    a. EnsemblID, UniprotID, UniparcID, RefseqID, pdbID, BiogridID, StringID, TaxonID
3. Gene ontology to find what processes the gene is associated with
    a. goID, goDescription, Keywords
4. DNA sequences and information for further downstream experiments (eg. For designing primers for subcloning or gene knockdown experiments)
    a. DNASeq, Strand, StartPos, EndPos
5. Protein sequences and information for further downstream experiments (eg. structure prediction if protein 3D structure is not elucidated via protein structure prediction methods such as AlphaFold, then for in silico drug discovery methods eg. Ligand binding etc.)
    a. ProteinSeq, ProteinFamily, ProteinLength, ProteinMass, ProteinResidueMod, ProteinPTM
6. Protein information to further guide researchers on choosing genes for downstream applications (eg. Deciding which proteins would be potential drug targets etc).
    a. ProteinFunction, ProteinFeatures, ProteinSubunits, ProteinDevStage, ProteinTissueLoci, ProteinSubCellLoci
7. Finding interactions which may suggest processes it is associated with, alongside any regulatory pathways that could be manipulated
    a. Interactions

## Some Features of the Pipeline

*Flexibility of pipeline*

The pipeline assumes none of the conditions and extracts unique conditions from the index Tco2.fq files. Therefore, additional samples can be added with more unique SampleType, Time and Treatment and the pipeline will still be able to run, provided the naming scheme of files and samples is the same.

*bowtie2 vs histat2*

Histat2 performs better than bowtie2 in alignment rate and gene coverage for long reads. However, it has been shown that when sequencing data is short and is using paired end reads, the alignment rates are very similar (Musich, Cadle-Davidson & Osier, 2021). Therefore, no preference was given for tool choice and bowtie2 was selected.

*Removing bad sequencing data*

A max number of fails from the fastqc quality check allowed for the sequencing data is set by the user to remove any samples with bad reads. This decreases the number data points which increases sampling bias, especially when there are only three replicates of each condition. However, including the samples with bad reads would lead to unreliable count data which affects the validity of the calculated fold changes.

*Fold change where reference = 0*

When calculating fold change, if the reference value is 0 the denominator is 0 leading to an inf value. To mitigate this, a small value was added to the reference when the reference value is 0. A log2 fold change, which is typical in calculating fold change of gene expression, was not done as the aim of the experiment is to elucidate alternative energy metabolism pathways. Therefore, the larger fold change will indicate more clearly the activation of such pathway.

## Tools Used

*fastqc* (Andrews, 2010), *bowtie2* (Langmead & Salzberg, 2012) (Langmead et al., 2019), *samtools* (Danecek et al., 2021), *bedtools* (Quinlan & Hall, 2010)

## Programme Parameters and Flags

*cp -u*

*-u* updates files and only copies newer or novel source files which reduces time taken for importing large amounts of data, especially when data files have been updated.

*fastqc -o –extract*

-*o* outputs files in specified directory. Done to organise working directory

-*extract* automatically extracts .zip folders for fastqc reports so downstream processes can access the summary.txt containing the pass/warn/fail data

*bowtie2-build -f*

-*f* sets input as fasta files. Done because input reference sequence is in fasta file format.

*bowtie2 -x -q -1 -2*

-*x* for index file

-*q* to set query input file format as .fq or .fastq. Done because input files are in .fq.gz file format.

-*1, -2* to set alignment for paired end reads, specifying which sequence data file inputs are end1 and end2.

*samtools view -b -o*

-*b* to output .bam files after converting the .sam files instead of outputting the default .sam files again.

-*o* to output in specified directory. Done to keep working directory organised.

*samtools sort –output-fmt BAM*

--*output-fmt BAM* to index and sort the .bam files so it can be read by *bedtools coverage*

*samtools index -b*

-*b* to generate .bai files

*bedtools coverage -counts -a -b*

-*counts* to give gene read counts

## <u>**User Manual**</u>

*Input data locations*

1. Place all RNAseq data within /localdisk/data/BPSM/ICA1/fastq/ OR place within ./tempdata directory in working directory.
2. Place *T. congolense* reference genome sequence within /localdisk/data/BPSM/ICA1/Tcongo_genome/ and .bedfiles containing information on gene locations in /localdisk/data/BPSM/ICA1/ OR place both in ./refseqdata/ in working directory.

3. chmod 700 all .sh files for permissions to execute

*Running the pipeline*

4. Run ./1getcounts.sh to start pipeline from beginning. Will automatically run next steps.
    a. ./2getmeans.sh will require input to specify threshold for fails in fastqc quality check in order to remove samples with low quality reads.
    b. ./3foldchange.sh will require input for query, reference, and range.

       Reference is what the query condition will compare against.
       ie. Reference is the denominator for fold change.

       Range refers to each unique value in a column to give a range of query conditions.
       eg. Reference condition = Clone1 0 Uninduced, Query condition = Clone2 0 Uninduced, Range = Time.

       Will output:
       Clone1 0 Uninduced vs Clone2 0 Uninduced
       Clone1 24 Uninduced vs Clone2 24 Uninduced
       Clone1 48 Uninduced vs Clone1 48 Uninduced

## Difficulties Experienced

Code is messy, with an over reliance on while read loops and temporary files. Some difficulties with syntax, as well as variables (since they do not automatically run the commands that are put in them in contrast to other languages).

Some difficulties also experienced with setting up the groupwise comparisons. Particularly selecting which conditions to compare against which conditions.

Though currently the pipeline is able to take any potential additional conditions ie. More rows and more unique SampleType, Time and Treatment, I unable to make the pipeline flexible enough to include any potential additional columns in the Tco2.fqfiles.

## Potential Additional/Alternative Features

*Trimming*

A large portion of the sequencing reads fail the 'per base sequencing content' test during the fastqc quality check, especially within the first 10-15 bases. This is typical of sequencing data due to factors such as primer binding (Crossley et al., 2020). However, due to the short reads of RNAseq data, as well as studies showing that trimming is not required for mapping RNAseq reads (Liao & Shi, 2020) and for paired end reads, little to no trimming results in the most accurate gene expression estimates (Williams et al., 2016), trimming was not performed.

Nonetheless, for more flexibility of the pipeline, the user should be able to specify if they want to trim, and if they want to trim each read a fixed amount or trim according to the quality of each base read.

*Multithread processing of alignments*

Currently, the pipeline's most time-consuming step is alignment. There is an option in bowtie2 to specify the number of threads to use. This was set as the default number since it is unknown the number of available threads for the user. Therefore, a potential addition would be to allow the user to specify how many threads they want to use for alignment.

*Using more powerful alignment tools*

Future versions of the pipeline may consider using alignment tools such as STAR or Kallisto. STAR is the next best contender as it has been shown to detect more unique gene counts than bowtie2 therefore STAR may be more useful since the aim of the experiment is to elucidate alternative energy metabolism pathways. Thus, differentially expressed genes that may have been undetected with bowtie2 may be detected with STAR. STAR and Kallisto also take a significantly shorter time to run than bowtie2 (Du et al., 2020).

*Normalization of count data*

The current count data that is produced is the raw counts of sequencing reads that map to a specific gene. The current method of comparing the raw counts is not valid due to the potential different transcript abundances between samples. To normalize and produce relative abundance measures, and thus gain valid differential gene expression levels, additional features can be added.

Within sample normalization, such as calculating the relative proportion of transcripts in the pool of RNA, TPM (Transcripts per million) , or calculating the fragments per kilobase of exon per million reads (FPKM) can be done to normalize the counts (Zhao, Ye & Stanton, 2020). However, since the main goal is to observe differential gene expression, the better method would be to calculate relative abundance measures between samples, such as with TMM (Trimmed Mean of M values) used by edgeR (Robinson, McCarthy & Smyth, 2010) or with the median of ratios used by DESeq2 (Love, Huber & Anders, 2014). However, this requires the use of housekeeping genes that are assumed to not be differentially expressed between experimental conditions, which may be erroneous if differentially expressed.

Thus, normalization of count data, particularly between samples, is a key additional feature to include for future versions of the pipeline.

*Incorrect assumption of no introns*

This pipeline assumes incorrectly that all genes have no introns. However, this is not the case as *T. congolense,* like most Trypanosoma, undergo trans splicing RNA events, where RNA is spliced between two different RNA molecules and therefore may come from multiple genes , in contrast to cis splicing RNA events in most eukaryotes where splicing only occurs within one RNA molecule therefore coming from only one gene (Michaeli, 2011).

Therefore, in future versions of the pipeline, tools which identify spliced leader trans splicing that occurs in Trypanosoma and predicts the operons which it originates from, such as SLIDR and SLOPPR (Wenzel, Müller & Pettitt, 2021), can be implemented.

## References

Andrews, S. (2010) No title. *FastQC: A Quality Control Tool for High Throughput Sequence Data.*

Crossley, B. M., Bai, J., Glaser, A., Maes, R., Porter, E., Killian, M. L., Clement, T. & Toohey-Kurth, K. (2020) Guidelines for Sanger sequencing and molecular assay monitoring. *Journal of Veterinary Diagnostic Investigation : Official Publication of the American Association of Veterinary Laboratory Diagnosticians, Inc.* 32 (6), 767-775. 10.1177/1040638720905833.

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M. & Li, H. (2021) Twelve years of SAMtools and BCFtools. *GigaScience.* 10 (2), giab008. 10.1093/gigascience/giab008.

Du, Y., Huang, Q., Arisdakessian, C. & Garmire, L. X. (2020) Evaluation of STAR and Kallisto on Single Cell RNA-Seq Data Alignment. *G3: Genes|Genomes|Genetics.* 10 (5), 1775-1783. 10.1534/g3.120.401160.

Langmead, B. & Salzberg, S. L. (2012) Fast gapped-read alignment with Bowtie 2. *Nature Methods.* 9 (4), 357-359. 10.1038/nmeth.1923.

Langmead, B., Wilks, C., Antonescu, V. & Charles, R. (2019) Scaling read aligners to hundreds of threads on general-purpose processors. *Bioinformatics.* 35 (3), 421-432. 10.1093/bioinformatics/bty648.

Liao, Y. & Shi, W. (2020) Read trimming is not required for mapping and quantification of RNA-seq reads at the gene level. *NAR Genomics and Bioinformatics.* 2 (3), lqaa068. 10.1093/nargab/lqaa068.

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. 10.1186/s13059-014-0550-8.

Michaeli, S. (2011) Trans-splicing in trypanosomes: machinery and its impact on the parasite transcriptome. *Future Microbiology.* 6 (4), 459-474. 10.2217/fmb.11.20.

Musich, R., Cadle-Davidson, L. & Osier, M. V. (2021) Comparison of Short-Read Sequence Aligners Indicates Strengths and Weaknesses for Biologists to Consider. *Frontiers in Plant Science.* 12 10.3389/fpls.2021.657240.

Quinlan, A. R. & Hall, I. M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics (Oxford, England).* 26 (6), 841-842. 10.1093/bioinformatics/btq033.

Robinson, M. D., McCarthy, D. J. & Smyth, G. K. (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics (Oxford, England).* 26 (1), 139-140. 10.1093/bioinformatics/btp616.

Wenzel, M. A., Müller, B. & Pettitt, J. (2021) SLIDR and SLOPPR: flexible identification of spliced leader trans-splicing and prediction of eukaryotic operons from RNA-Seq data. *BMC Bioinformatics.* 22 (1), 140. 10.1186/s12859-021-04009-7.

Williams, C. R., Baccarella, A., Parrish, J. Z. & Kim, C. C. (2016) Trimming of sequence reads alters RNA-Seq gene expression estimates. *BMC Bioinformatics.* 17 (1), 103. 10.1186/s12859-016-0956-2.

Zhao, S., Ye, Z. & Stanton, R. (2020) Misuse of RPKM or TPM normalization when comparing across samples and sequencing protocols. *RNA (New York, N.Y.).* 26 (8), 903-909. 10.1261/rna.074922.120.

Lang, D. T. (2023) *RCurl: General Network (HTTP/FTP/...) Client Interface for R.* https://CRAN.R-project.org/package=RCurl.

Martin, F. J., Amode, M. R., Aneja, A., Austine-Orimoloye, O., Azov, A. G., Barnes, I., Becker, A., Bennett, R., Berry, A., Bhai, J., Bhurji, S. K., Bignell, A., Boddu, S., Branco Lins, P.,R., Brooks, L., Ramaraju, S. B., Charkhchi, M., Cockburn, A., Da Rin Fiorretto, L., Davidson, C., Dodiya, K., Donaldson, S., El Houdaigui, B., El Naboulsi, T., Fatima, R., Giron, C. G., Genez, T., Ghattaoraya, G. S., Martinez, J. G., Guijarro, C., Hardy, M., Hollis, Z., Hourlier, T., Hunt, T., Kay, M., Kaykala, V., Le, T., Lemos, D., Marques-Coelho, D., Marugán, J. C., Merino, G. A., Mirabueno, L. P., Mushtaq, A., Hossain, S. N., Ogeh, D. N., Sakthivel, M. P., Parker, A., Perry, M., Piližota, I., Prosovetskaia, I., Pérez-Silva, J.,G., Salam, A. I. A., Saraiva-Agostinho, N., Schuilenburg, H., Sheppard, D., Sinha, S., Sipos, B., Stark, W., Steed, E., Sukumaran, R., Sumathipala, D., Suner, M., Surapaneni, L., Sutinen, K., Szpak, M., Tricomi, F. F., Urbina-Gómez, D., Veidenberg, A., Walsh, T. A., Walts, B., Wass, E., Willhoft, N., Allen, J., Alvarez-Jarreta, J., Chakiachvili, M., Flint, B., Giorgetti, S., Haggerty, L., Ilsley, G. R., Loveland, J. E., Moore, B., Mudge, J. M., Tate, J., Thybert, D., Trevanion, S. J., Winterbottom, A., Frankish, A., Hunt, S. E., Ruffier, M., Cunningham, F., Dyer, S., Finn, R. D., Howe, K. L., Harrison, P. W., Yates, A. D. & Flicek, P. (2023) Ensembl 2023. *Nucleic Acids Research.* 51 D933-D941. 10.1093/nar/gkac958.

Nightingale, A., Antunes, R., Alpi, E., Bursteinas, B., Gonzales, L., Liu, W., Luo, J., Qi, G., Turner, E. & Martin, M. (2017) The Proteins API: accessing key integrated protein and genome information. *Nucleic Acids Research.* 45 W539-W544. 10.1093/nar/gkx237.

Ooms, J. (2023) *curl: A Modern and Flexible Web Client for R.* https://CRAN.R-project.org/package=curl.

Ooms, J. (2014) *The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects.*

Ooms, J., James, D., DebRoy, S., Wickham, H. & Horner, J. (2023) *RMySQL: Database Interface and 'MySQL' Driver for R.* https://CRAN.R-project.org/package=RMySQL.

Oughtred, R., Rust, J., Chang, C., Breitkreutz, B., Stark, C., Willems, A., Boucher, L., Leung, G., Kolas, N., Zhang, F., Dolma, S., Coulombe-Huntington, J., Chatr-Aryamontri, A., Dolinski, K. & Tyers, M. (2021) The BioGRID database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions. *Protein Science : A Publication of the Protein Society.* 30 (1), 187-200. 10.1002/pro.3978.

The UniProt Consortium. (2023) UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research.* 51 D523-D531. 10.1093/nar/gkac1052.

Voisinne, G. (2019) *queryup: Query the UniProt REST API using R.* Centre d'Immunologie de Marseille-Luminy, Aix Marseille Université, INSERM, CNRS, Marseille, France. https://github.com/VoisinneG/queryup/.

Wickham, H. (2023) *httr: Tools for Working with URLs and HTTP.*

Yates, A., Beal, K., Keenan, S., McLaren, W., Pignatelli, M., Ritchie, G. R. S., Ruffier, M., Taylor, K., Vullo, A. & Flicek, P. (2015) The Ensembl REST API: Ensembl Data for Any Language. *Bioinformatics.* 31 (1), 143-145. 10.1093/bioinformatics/btu613.