XPRESSyourself: Automating and Democratizing High-Throughput Sequencing

Jordan A. Berg,[1] Jeffrey T. Morgan,[1] Jonathan R. Belyeu,[2] Alex J. Bott,[1] Yeyun Ouyang,[1] Aaron R. Quinlan,[2,4,5] Jason Gertz,[3] Michael T. Howard,[2] Jared P. Rutter[1,6*]

[1]Department of Biochemistry, University of Utah, Salt Lake City, UT, USA, 84112
[2]Department of Human Genetics, University of Utah, Salt Lake City, UT, USA, 84112
[3]Department of Oncological Sciences, University of Utah, Salt Lake City, UT, USA, 84112
[4]USTAR Center for Genetic Discovery, University of Utah, Salt Lake City, UT, USA, 84112
[5]Department of Biomedical Informatics, University of Utah, Salt Lake City, UT, USA, 84112
[6]Howard Hughes Medical Institute, University of Utah, Salt Lake City, UT, USA, 84112

*To whom correspondence should be addressed; E-mail: rutter@biochem.utah.edu.

Sequencing experiments are routine and often necessary in biological and clinical research. However, for the average user, a computational overhead often exists. XPRESSyourself is a ribosome profiling and general RNA-seq pipeline that aims to eliminate these barriers, standardize in silico protocols, and decrease time-to-discovery from data. With these tools, a user can go from raw data to publication quality figures in a matter of hours. Additionally, XPRESSyourself introduces tools missing from the ribosome profiling and RNA-seq computational toolkit. Using XPRESSyourself, we discovered new putative hits from publicly available ribosome profiling data, highlighting its utility to reveal biological insight.

Keywords

Pipeline, Ribosome Profiling, RNA-seq, Automation, Standardization

1 Background

High-throughput profiling of gene expression data has revolutionized biomedical, industrial, and basic science research. Within the last two decades, RNA-seq has found itself the forerunner technology for high quality gene expression profiling, as it can measure relative transcript abundance, differential splice variants, sequence polymorphisms, and more (*1*). This technology has been co-opted to create a variety of technologies such as single-cell RNA-seq, ChIP-seq, and ribosome profiling (*2*).

While vast strides have been made to these technologies, various bottlenecks still exist. For example, while more and more researchers are becoming accustomed to to the field of bioinformatics and computational biology, learning the intricracies of the different tools used in processing RNA-seq data can be inhibitory if users are not aware of the proper tools to use or use outdated software (*3, 4*). Even for the experienced user, developing robust, automated pipelines that meticulously process and assess quality of these datasets can be laborious, notwithstanding the induced variability of multiple pipelines.

While several computational pipelines for sequencing have emerged intending to tackle various aspects of these bottlenecks, many suffer from usability issues, are not easily modifiable, or sacrifice quality for speed. While RNA-seq is a more matured technology, there are still an abundance of biases and idiosyncrasies associated with each method or tool of which the beginner user may not be aware. Additionally, few if any pre-existing pipeline or

toolkit offer a thorough set of integrated tools for handling common quality control issues or reference creation. For example, a common bias in ribosome profiling libraries is a 5' transcript pile-up (*5–7*). It is recommended that this region of each transcript not be quantified when processing ribosome profiling libraries; however, currently no tools exist to facilitate this essential step (*8, 9*).

In response to the issues surrounding the automation and democratization of sequencing technology, we designed the XPRESSyourself bioinformatics suite for processing and analyzing high-throughput expression data. Architectually, this suite is designed to work fast (1.48 min per gigabyte of raw input data), while not sacrifices quality for speed. Each step of the pipelines utilize the state of the art software package for that task, having been previously vetted by peer-reviewed benchmarking studies. Additionally, the scaffold that creates this pipeline structure is designed in a way that update and testing of a new module when better software comes along should only take a couple of hours.

With the XPRESSyourself package XPRESSpipe, the user is provided with a complete suite of software to handle pre-processing, aligning, and quantifying reads, performing quality control via various meta-analyses of pre- and post-processed reads. We also provide access to key quality control measures useful for assessing ribosome profiling and other RNA-seq experiments. These include read length distributions to ensure correct sequencing library read sizes and a periodicity sub-module that tracks the P-site of ribosome footprints to assess effective capture of the ribosome's characteristic 1 codon step. These measurements are particular helpful for ribosome profiling experiments due the unique characteristics of the libraries. XPRESSpipe also includes a metagene analysis sub-module that shows the distribution of all aligned reads across a representative transcript a library and a library complexity visualization sub-module to ensure minimalization of PCR duplicates. Additionally, housed within the XPRESSyourself XPRESSplot package, tools are provided to perform the bulk of sequence analysis and generation of figures for publication, where many plot generation protocols that might take several hundred lines of code are whittled down to one line. XPRESSyourself suite tools are freely available under a GPL-3.0 license at https://github.com/XPRESSyourself

2 Results

2.1 XPRESSpipe

XPRESSpipe pipelines have been designed for ribosome profiling and other single-end and paired-end RNA-seq. Each pipeline offers a handful of tunable parameters to the user, while keeping most parameters hidden to the benefit of the average user. The pipeline requirements are also largely based upon The Cancer Genome Atlas (TCGA) (https://www.cancer.gov/tcga) alignment standards and ensure standardization of alignment. In the future it is feasible that additional tunable parameters will be added. For the purposes of this manuscript, we will focus on ribosome profiling examples, while the majority of statements are also applicable to single- and paired-end RNA-seq. More details can be found in the documentation (https://xpresspipe.readthedocs.io/en/latest/). Table 1 outlines these parameters.

Table 1: Summary of XPRESSpipe pipeline arguments.

Arguments	Description
Required	
`-i, --input`	Path to input directory
`-o, --output`	Path to output directory
`-r, --reference`	Path to parent organism reference directory
`-g, --gtf`	Path and file name to GTF used for alignment quantification
`-e, --experiment`	Experiment name
Optional	
`--two_pass`	Include option to perform a two-step alignment to map for unannotated splice-juntions
`--mask`	Include option to perform a primary masking alignment step to remove unwanted, often highly repetative sequences
`-a, --adaptors`	Specify adaptor as string – if "None" is provided, software will attempt to auto-detect adaptors – if "POLYX" is provided as a single string in the list, polyX adaptors will be trimmed
`-q, --quality`	PHRED read quality threshold (default: 28)
`--min_length`	Minimum read length threshold to keep for reads (default: 18)
`--count_duplicates`	Include option to quantify alignment files without de-duplication
`--output_bed`	Include option to output BED files for each aligned file
`--output_bigwig`	Include flag to output bigwig files for each aligned file
`-c, --quantification_method`	Specify quantification method (default: Cufflinks (*10*))
`--method`	Normalization method to perform (options: "RPM", "TPM", "RPKM", "FPKM")
`--batch`	Include path and filename of dataframe with batch normalization parameters
`--sjdbOverhang`	Sequencing platform read-length for constructing splice-aware reference previously (see STAR documentation for more information)
`--mismatchRatio`	Alignment ratio of mismatches to mapped length is less than this value (see STAR documentation for more information)
`--seedSearchStartLmax`	Adjusting this parameter by providing a lower number will improve mapping sensitivity (recommended value = 15 for reads 25 nts) (see STAR documentation for more information)
`-m, --max_processors`	Number of max processors to use for tasks (default: No limit)

2.1.1 Installation

XPRESSpipe can be compiled from source (https://github.com/XPRESSyourself/XPRESSpipe) or a version-controlled Docker image (https://www.docker.com/) can be loaded using the following commands:

Listing 1: Source installation.

```
Download and unzip archived version from https://github.com/XPRESSyourself/XPRESSpipe/releases
$ cd XPRESSpipe
$ python setup.py install
```

XPRESSpipe is built upon several pre-established software packages required as dependencies. A full list can be found in the Methods. The source install will automatically check the system for Anaconda (*11*) and install required dependencies. Docker images come with these dependencies pre-installed.

Listing 2: Docker installation

```
$ docker image pull jordanberg/xpresspipe:latest
```

2.1.2 Inputs

While inputs will vary sub-module to sub-module, and further information can be found in the documentation (https://xpresspipe.readthedocs.io/en/latest/) or by entering `xpresspipe <sub-module name> --help`, a few points of guidance are important to consider.

- Single-end reads should end in `.fa`, `.fasta`, or `.txt`
- Paired-end reads should end in `.read1/2.fa` or `.r1/2.fa`, where `.fa` could also be `.fasta` or `.txt`

- The transcriptome reference file should be a valid GTF file and should be named `transcripts.gtf`
- If specifying a group of fasta files to use for alignment or reference curation, the directory containing these files cannot contain any other files ending in `.txt` or `.fa`

2.1.3 Reference Curation

One of the first steps of RNA-seq alignment is curating a reference for the alignment software to map reads. For the purposes of the current version of XPRESSpipe, a STAR (*12*) reference should be created. An Ensembl-formatted (https://ensembl.org) GTF should also be placed in the reference directory and be named `transcripts.gtf`. Additional modifications are recommended to this file, which can be performed using this sub-module, discussed in more detail below. Additionally, any chromosome `.fasta` files should be places in their own directory within the curated reference directory. As this can be a time-consuming process, we will leave the `--max_processors` argument as default in order to utilize all cores available to the computing unit. Additionally, if a masking step will be required of the pipeline, the appropriate `.fasta` file or files should be places in their own directory and the

Listing 3: curateReference example

```
$ xpresspipe curateReference -o /path/to/output/location/ \
                             -f /path/to/fasta/genome/ \
                             -g /path/transcripts.gtf \
                             --masked_index /path/to/fasta/mask/ \
                             --protein_coding \
                             --longest_transcript \
                             --truncate \
                             --truncate_5prime 45 \
                             --truncate_3prime 15 \
                             --sjdbOverhang 49 \
                             --max_processors None
```

2.1.4 GTF Modification

As ribosomal RNAs and other non-coding RNAs are highly abundant in RNA-seq experiments, it is often recommended to disregard these sequences. While the masking step mentioned above can handle this issue, it is also recommended to do so during the quantification step. By providing the `--protein_coding` argument, only protein-coding genes are retained in the GTF file. In most eukaryotes, mRNAs undergo alternative splicing. However, some quantification will consider the multiply annotated splice variants of a gene as a multi-mapper since they map to a location where several isoforms of the same gene overlap. These reads are either penalized or discarded. By providing the `--longest_transcript` argument, the longest coding transcript for each gene is retained in the GTF file. For the purposes of ribosome profiling, where 5' and 3' transcript biases are frequent (*8,9*), the 5' and 3' ends of each transcript record need to be trimmed to avoid quantification to this region. By providing the `--truncate` argument, the 5' and 3' ends of each transcript will be trimmed by the specified amounts.

2.1.5 Read Processing

While all intermediate steps of the pipelines can be run singly, we will describe the outline of the software in the context of the ribosome profiling pipeline. Pipelines and individual sub-modules are capable of being run in a parallel manner for each input file, thus accelerating the overall process. Descriptions of the options can be found in Table 1.

1. **Trimming**: First, reads need to be cleaned of artifacts from library creation. These include adaptors, unique molecular identifier (UMI) sequences, and technical errors in the form of low quality base calls. By doing so, non-native sequences are removed and reads can align properly to the reference. XPRESSpipe uses fastp, a faster, more accurate trimming package that has improved alignable read output (*13*). Adaptor sequence, base quality, and read length are all adjustable parameters available to the user. Additionally, features, such as UMIs can be input and used in pre-processing to remove artifacts from PCR duplication (*14*).

2. **Alignment**: After trimming, reads are then aligned to a reference genome. XPRESSpipe uses STAR, which, while being a more memory intensive approach, is fast and one of the best performing sequence alignment options currently available (*12, 15*). XPRESSpipe is capable of performing a masking alignment, a single-pass, splice-aware GTF-guided alignment, or a two-pass alignment of reads wherein novel splice junctions are determined and built into the reference, followed by alignment of reads to the new reference. A sorted-by-coordinate and indexed BAM file by STAR.

3. **Post-alignment Processing**: XPRESSpipe will further process alignment files by parsing files for only unique alignments that are then passed on to the next steps. PCR duplicates are detected and marked or removed for downstream processing; however, these files are not used in cases where UMIs were provided or where the user specifies to use un-deduplicated alignments for downstream processing. These steps are performed using samtools (*16*). Optionally, BED and bigWig files can also be output. These conversions are handled by bedtools (*17*) and deeptools (*18*).

4. **Read Quantification**: XPRESSpipe quantifies read alignments for each input file using Cufflinks (*10, 19*). If a modified GTF was provided as input, it is used here to avoid potential pitfalls with read quanitification (i.e. a read being counted as a multi-mapper when belonging to multiple transcripts for the same gene). Optionally, a user may use HTSeq (*20*) so that quantification behavior conforms to current TCGA standards involving a strand agnostic, `intersection-nonempty` protocol. However, use of HTSeq is not recommended due to documented short-comings (*19*).

5. **Normalization**: Methods for count normalization are available within XPRESSpipe by way of the XPRESSplot package described later. For normalizations involving transcript length, the appropriate GTF must be provided. For samples sequenced on different chips, prepared by different individuals, or on different days, the `--batch` argument should be provided along with the appropriate metadata matrix, which is then processed through XPRESSplot (*21*). Sample normalization methods available include reads-per-million (RPM), Reads-per-kilobase-million (RPKM) or Fragments-per-kilobase-million (FPKM), and transcripts per million (TPM) normalization (*22*).

6. **Quality Control**: It is important to perform quality control of sequencing samples to ensure the interpreted results are reliable. XPRESSpipe performs a variety of quality control measures. For each analysis type, high-resolution, publication quality summary figures are output for all samples in a given experiment for quick reference to the user.

    - **Read Length Distribution**: Per sample, the lengths of all reads are analyzed by FastQC (*23*) after trimming. By assessing the read distribution of each sample, the user can ensure the expected read

size was sequenced. This is particularly helpful for ribosome profiling experiments as insertion of the requisite 21-30 nt ribosome footprints into the sequencing library was successful (*8*).

- **Library Complexity**: Analyzing library complexity is an effective methods for analyzing the robustness of a sequencing experiment of capturing various mRNA species. As the majority of RNA-seq preparation methods involve a PCR step, at times certain fragments are favored and over-replicated in contrast to others. By plotting the number of PCR replicates versus expression level, one can determine how successful the library preparation was at reducing these biases and at capturing a robust subset of mRNAs. The analysis is performed using dupRadar (*24*) where inputs are the PCR duplicate-tagged BAM files output by XPRESSpipe. Duplicate tagging is performed by samtools (*16*).
- **Metagene Estimation Profile**: In order to identify any general 5' or 3' biases in captured transcripts, a metagene profile can be created for each sample. This is performed by determining the meta-genomic coordinate for each aligned read in exon space. Required inputs are an indexed BAM file and an unmodified GTF reference file and outputs are metagene metrics, individual plots, and summary plots.
- **Codon Phasing/Periodicity Estimation Profile**: In ribosome profiling, a useful measure of a successful experiment comes by investigating the codon phasing of ribosome footprints (*8*). To do so, the P-site is calculated for each mapped ribosome footprint by taking the genomic coordinate 16 nucleotides upstream of the 3' end of each transcript and measuring the distance in nucleotides along exon space to the start of the transcript (*25*). The same inputs are required as for the `metagene` sub-module, and the penalty is calculated in the same manner. This method if intended as a quality control and will provide a good estimate of codon phasing in ribosome profiling data. However, it does forgo any further normalization, therefore it may not be best suited for more in-depth studies of codon phasing dynamics.
- **rRNA Depletion Probe**: Ribosomal RNA (rRNA) contamination is common in RNA-seq library preparation and the bulk of RNA in a cell at any given time is dedicated to rRNA. As unique rRNA sequences are relatively few and therefore highly repeated in sequencing libraries without depletion. Depletion of these sequences is often desired in order to have better depth of coverage of mRNA sequences. In order to facilitate this depletion, many commercial kits are available that target specific rRNA sequences for depletion, or that enrich mRNA polyA tails. However, and especially in the case of ribosome profiling experiments, where RNA is digested by an RNase to create ribosome footprints, many commercial depletion kits will not perform sufficiently and polyA selection kits are inoperable as footprints will not have the requisite polyA sequence. To this end, custom rRNA probes are recommended (*2, 8*). `rrnaProbe` will compile over-represented sequences between footprint libraries within the overall experiment and output a rank ordered list of these sequences for probe design.

2.1.6 Outputs

While outputs will vary sub-module to sub-module, generally, the user will specify a parent output directory and necessary sub-directories will be created based on the step in the pipeline. Further information can be found in the documentation (https://xpresspipe.readthedocs.io/en/latest/) or by entering `xpresspipe <sub-module name> --help`. Figure 1 provides an example of the output file scheme for XPRESSpipe. For a complete pipeline

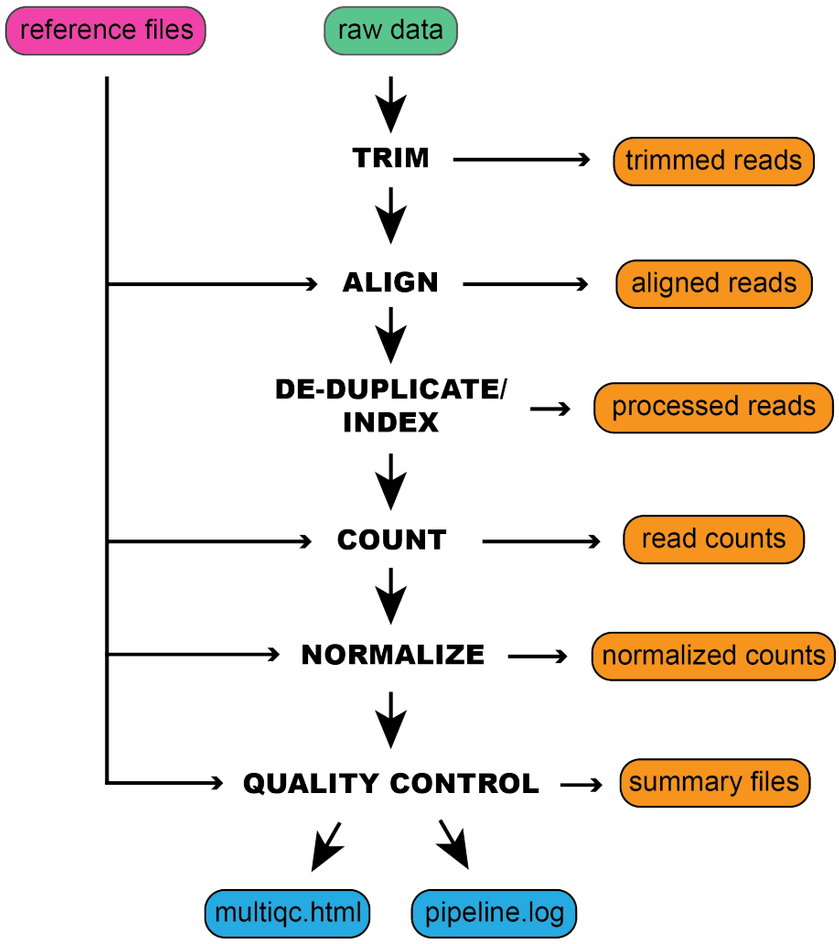

Figure 1: An example schematic of the inputs required by XPRESSpipe and organization of the ouputs.

run, the user can expect BAM alignment files, a collated count table of all samples in the experiment, and quality control figures. For almost all sub-modules, a log file will also be output to track performance and possible errors.

2.2 XPRESSplot

Further analysis of ribosome profiling or RNA-seq data is handled within XPRESSplot. XPRESSplot is a python library of analysis and plotting tools that builds upon existing packages, such as Matplotlib (*26*) and Seaborn (*27*) to generate flexible, specific analyses and plots frequently used by biological researchers that can each be executed in a single line of code rather then tens to hundreds. Additionally, many included analytical features are currently available in an R package but not in a Python package, a programming language becoming more and more common in biological research. A summary of new or more automated tools is provided below and methods are discussed in subsequent sections. We refer the reader to the documentation (https://xpressplot.readthedocs.io/en/latest/?badge=latest) for more details instructions for other features currently in the toolkit, as well as for future features to be added. XPRESSplot is also capable of handling other omics datasets in many cases.

2.2.1 Getting Data

Generally, two inputs are required for all functions within XPRESSplot:

1. **Expression Matrix**: It is assumed the input data matrix = *i* * *j* where *i* (columns) are samples and *j* (rows) are genes or other relative measurement points.

2. **Metagene Table**: It is assumed the metagene table is a two column, header-less data matrix where column 0 is the sample ID (as specified in *i* of the expression matrix) and column 1 is the sample group (for example, wild-type or treatment).

2.2.2 Normalization

RNA-seq experiments can be normalized using the reads-per-million (RPM), Reads-per-kilobase-million (RPKM) or Fragments-per-kilobase-million (FPKM), and transcripts per million (TPM) methods, as outlined in Equations 1-4 in the Methods (*22*). Other normalizations, such as mean centering of *j* axis components (i.e. genes or other items) by sklearn's preprocessing module (*28*). Count thresholds can also be set to remove genes from analysis that may be less reliable due to poor ability to be sequenced.

2.2.3 Analyzing Data

While a litany of analysis tools are included in XPRESSplot as of the time of writing, we will focus on tools unique to this Python library or particularly useful and refer to reader to the documentation for further details and examples of additional analysis features, current and future.

- **Principle Components Analysis**: Principle components analysis (PCA) for the data matrix is computed using Python's scikit-learn package (*28*) and desired principle components are plotted in a scatter plot via the matplotlib (*26*) and seaborn (*27*) packages. The XPRESSplot PCA module, as in many other analysis modules within XPRESSplot, samples are color-coded by cross-referencing the data matrix with the metagene table to determine sample label. A dictionary is additionally passed into the function that maps a particular color to each sample label. Confidence intervals are plotted over the scatterplot using numpy (*29, 30*).

- **Volcano Plot**: Volcano plots are an efficient method for plotting magnitude, direction, and significance of changes in expression or other data types between two conditions with multiple replicates each. By providing the categorical names for samples of two conditions in the metadata matrix, XPRESSplot will automate the calculation and plotting of this plotting method. For each gene, expression levels are averaged between the two conditions and the $\log_2$(fold change) is calculated. Additionally, for each gene, the P-value between the two conditions is calculated using scipy's individual T-test function (*31*). The $\log_2$(fold change) and -$\log_{10}$(P-value) is then plotted for each gene between the two conditions. Additional features available are the ability to plot threshold lines, highlight subsets of genes within the plot, and label specific genes by name.

- **Differential Expression Analysis**: XPRESSpipe includes a Python wrapper for DESeq2 for performing differential expression analysis of count data. We refer users to the original publication for more information about uses and methodology (*32*).

2.3 Validation

In order to evaluate the ability of XPRESSpipe to provide the user with reliable results, we processed publicly available raw sequence files through the pipeline. We chose to highlight a ribosome profiling dataset and a subset of TCGA samples to showcase the utility of XPRESSpipe for rapidly gleaming interesting molecular patterns and insights from publicly available data.

2.3.1 Ribosome Profiling Data and New Insights from Old Data

The integrated stress response (ISR) is a signaling mechanism used by cells and organisms due to a variety of cellular stresses and has been associated with a variety of diseases. Of particular interest, many disorders

resulting in neurological decline are associated with the ISR (*33*). While acute ISR is essential for proper cell survival, long periods of sustained ISR can be damaging. A recently discovered molecule, ISRIB, has recently come under the spotlight for its therapeutic potential and relative lack of side-effects. Interestingly, ISRIB is able to suppress low levels of ISR, while seemingly uneffective at managing acute, high-levels of ISR. It has also been shown to be neuroprotective in mouse models of acute neurological damage (*34–37*).

A recent study (GSE65778) utilized ribosome profiling in order to better understand the mechanisms of ISRIB and ISR, modeled by tunicamycin (Tm) treatment, from a protein translation viewpoint (*36*). Some key findings from this study were that during acute ISR, a specific subset of mRNAs were translationally regulated, and that canonical signaling factors were part of this response. In order to showcase the utility of XPRESSpipe in analyzing ribosome profiling and sequencing datasets, we re-processed and analyzed this dataset using more current in silico techniques included in the XPRESSpipe package to shed more light on the mechanisms of ISR and ISRIB mode of action. To process the 32 raw files from GEO, using 16 cores, it took 06h33m42s in wall-clock time, or 1d03h18m27s in CPU time. Compared to the raw count data made available in the original manuscript, samples showed comparable alignment rates (Spearman R values ranging from 0.710-0.865) (Figure 2A) as, according to the methods of the original paper, a now outdated alignment program, TopHat2 (*38*), was used that has a documented higher false positive alignment rate compared to STAR (*12, 39*). Processing of replicates within the XPRESSpipe method showed excellent correlation (Spearman R values all $>$0.99) (Figure 2B). Additionally, samples appeared to be relatively comparable to one another once RPM normalization was performed (Figure 2C).

The XPRESSpipe method interestingly showed that even more limited mRNA regulation was able to occur within the 1hr acute Tm treatment than previously appreciated (Figure 3A), as indicated by the constrained range high-confidence (FDR $<$0.01) points fell into along the X-axis, while the range is comparable for translation regulation. Similar canonical targets of translational regulation during ISR were identified as in the original study, such as ATF4 and ATF5 (Figure 3A, highlighted in magenta) (*36*). Other targets highlighted in the original study (*36*), either narrowly missed the FDR or fold-change cut-offs or had a handful of samples with 10-25 counts for the given gene, thereby missing the stricter count cut-off, such as DDIT3 (or CHOP; fold-change = 2.09, FDR = 0.037) and PPP1R15A (or GADD34; fold-change = 1.95, FDR = 0.013). Interestingly, using these methods and thresholds, the subset of translationally down-regulated transcripts during simulated ISR all possess a neurodegenerative phenotype based on a literature search (descriptions sourced from www.genecards.org and www.omim.org) (Table 2). This might be helpful in explaining the neurodegenerative phenotype associated with ISR. When an ISR model is treated with ISRIB, the translation efficiency levels of each of these genes returns to a more wildtype-like state (Figure 3B). Further analysis of these potential hits is strengthened by looking at the read pile-ups in IGV (*40*), where footprint coverage is observed across the transcript for all ribosome profiling samples except the Tm-treated ones (Figure 3C). This provides further strength to this claim as treatment with CircLigase in the library preparation can bias certain reads' incorporation in sequencing libraries (*41*).

Table 2: Translationally down-regulated genes during acute Tm treatment.

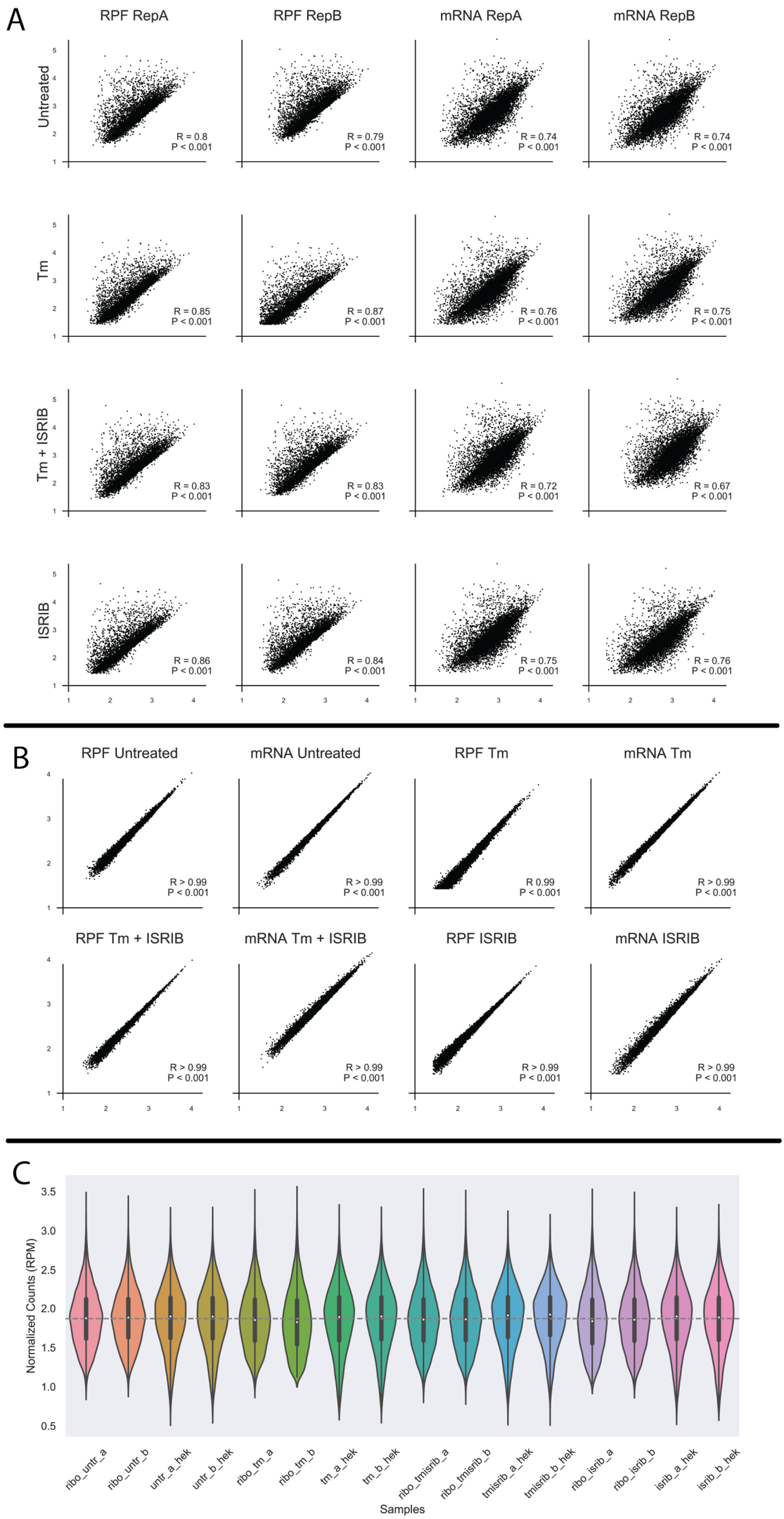

Figure 2: Broad sample quality control and comparision to original processing. A) Cross-processing comparisions between original manuscript and XPRESSpipe. B) Intra-processing comparisions between replicates for XPRESSpipe processing. C) Sample gene count distributions. Boxplots embedded within violinplot designate the interquartile range and violinplot reports count density. Note: All R values reported are Spearman R values.

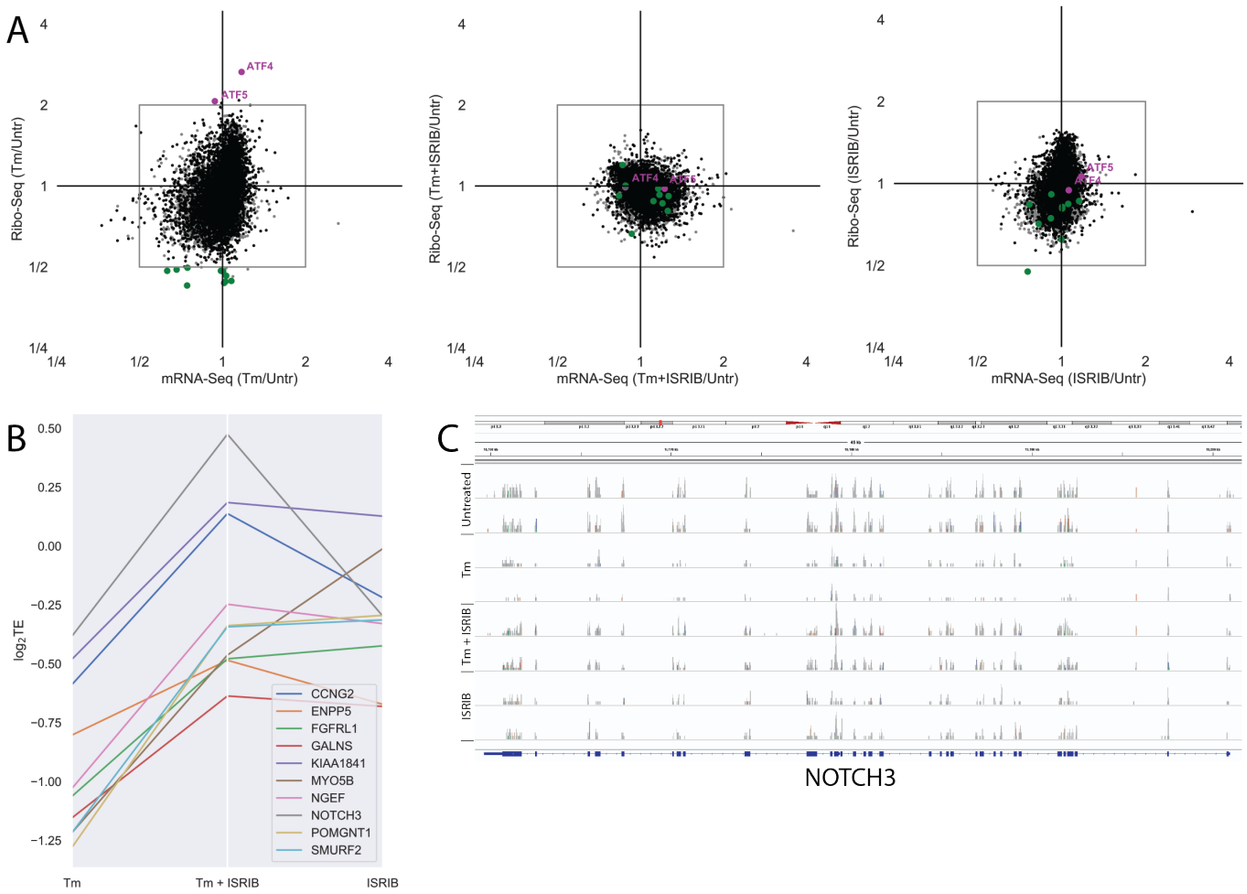

Figure 3: Biological validation and insight into previously published ISR model ribosome profiling data. A) Fold change for each drug condition compared to untreated for the ribosome profiling and RNA-seq data. ISR canonical targets passing fold change and significance threshold during Tm conditions are highlighted in magenta. All translationally down-regulated genes passing fold change and significance threshold during Tm conditions are highlighted in green. B) Changes in $\log_2$ Translation Efficiency compared to Untreated for each drug condition for each of the translationally down-regulated genes passing previous discussed thresholds. C) IGV plot for NOTCH3 for all ribosome profiling samples show read coverage across the transcript.

Gene Name	Relevant Description
CCNG2	Overexpressed in brain; Tightly regulated in cell cycle
ENPP5	Suggested role in neuronal cell communications in rat studies
FGFRL1	Highly expressed in brain; Sets in motion a cascade of downstream signals ultimately influencing mitogenesis and differentiation; Behavioral and neurological phenotypes
GALNS	Involved in breaking down and recycling molecules in lysosome
KIAA1841	Identified in a screen of proteins expressed in brain
MYO5B	Identified in a screen for large proteins expressed in brain; Encodes molecular motor that utilizes energy from ATP hydrolysis to generate mechanical force, required for the recycling of transferrin receptor back to the plasma membrane through an endocytotic recycling compartment in nonpolarized cells.
NGEF	Overexpressed in brain; Involved in axon guidance regulating ephrin-induced growth cone collapse and dendritic spine morphogenesis
NOTCH3	Establishes an intercellular signalling pathway that plays a key role in neural development; Mutations in NOTCH3 have been identified as the underlying cause of cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy
POMGNT1	Mutations in this gene may be associated with muscle-eye-brain disease and several congenital muscular dystrophies
SMURF2	Interacts with SMAD proteins; Functions in regulation of neuronal and planar cell polarity, induction of senescence, and tumor suppression

2.3.2 Performance Validation Using TCGA Data

In this section, we will process the raw data for 10-20 TCGA samples and compare the FPKM counts output by XPRESSpipe to the publicly available FPKM counts for the relevant samples. This will a simple section with one supplemental figure showing comparisons and probably end up being added right before bioRXiv submission. Summarize metrics (total reads mapped, etc)

2.3.3 Cost Analysis

XPRESSpipe functions can be computationally intensive and thus super-computing resources are recommended. Many universities provide super-computing resources to their staff; however, in cases where these resources are not available, servers such as Amazon's AWS can be used to process sequencing data using XPRESSpipe. For example, the ribosome profiling dataset used in the study was processed using the University of Utah's Center for High Performance Computing resources. Run statistics can be found in Table **??**.

Table 3: XPRESSpipe processing statistics for dataset GSE65778.

Metric	**Value**
Elapsed Real Time	06h19m55s
Total CPU Time	2d07h52m58s
Allocated CPUs	16
Allocated Memory per Node	62.50GB
Maximum RAM of all tasks	5.10GB

2.3.4 Comparisons to Other Workflows

Many instances of RNA-seq pipelines can be found from simple internet search but each have serious limitations. The first class is a tutorial titled a pipeline. Many instances of these can be found (https://www.encodeproject.org/rna-seq/, https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/); however, they are not automated and are often outdated. The second class is an automated pipeline, but requires extensive manual configuration (https://github.com/PavlidisLab/rnaseq-pipeline, https://github.com/nf-core/rnaseq, https://github.com/UMCUGenetics/RNASeq, https://github.com/cellgeni/rnaseq). The second class is an automated pipeline, but requires programmatic configuration to modify common parameters (https://github.com/dnanexus/tophat_cufflinks_rnaseq, https://www.nextflow.io/example4.html). Perhaps, the most user friendly case is Galaxy, but in cases like the ribosome profiling pipeline, methods are severly outdated and a robust quality control step is lacking. In all cases, a thorough, robust, simple pipeline geared to the general user without sacrificing for speed or quality is lacking.

3 Discussion

We have described herein a new software suite, XPRESSyourself, a collection of tools and pipelines to aid in expression data processing and analysis. While RNA-seq technologies are becoming more and more mature, standardized protocols are lacking. This is problematic when individuals are groups may not be using the most up-to-date methods or be aware of particular biases or measures of quality control required to produce a reliable, high-quality sequencing study. XPRESSpipe handles these issues for the user by continuously updating to utilize the best-performing software tools in sequencing as measured by peer-reviewed benchmarking studies. It also outputs all necessary quality control metrics so that the user can quickly assess quality and identify any systematic biases that may be present in their samples.

An additional problem XPRESSpipe addresses is the ability to use these software tools and be aware of proper execution of said tools, especially for those coming from a non-computational background. This is especially important in situations where a bioinformatics core may be so backlogged it takes 2-4 months to process a dataset, which is inhibitory to scientific discovery and dissemination. XPRESSyourself will dissolve this barrier

to entry for most users so that they can process and analyze their data immediately upon receipt of the raw data and only requires simple programming knowledge covered by a variety of free online programs (such as https://www.codecademy.com/learn/learn-the-command-line).

Tools previously missing from the general ribosome profiling toolkit have also been added within XPRESSyourself. This includes GTF truncation of transcripts in a recursive manner over exon space and rRNA probe design aids for removing contaminating rRNA sequences in ribosome profiling libraries that are difficult to remove with commercial kits.

We showed the utility of the XPRESSyourself toolkit by re-analyzing a publicly available ribosome profiling dataset. From this analysis, we identified putative hits that may contribute to the neurodegenerative effects of integrated stress response (ISR) and how the molecule ISRIB may be acting on these hits to act as a neuroprotective agent. This additionally highlights the importance of re-analyzing older datasets with more current methods, as over time methodologies improve to reduce false positives or negatives and increase mappability overall. We also showed the capability of quickly processing data on TCGA data. These principles are transferable to new datasets and XPRESSyourself will have individuals and labs take sequence processing and analysis into their own hands to avoid long queues with bioinformatics cores, save money, and overall democratize the process.

While XPRESSyourself's pipeline may be more time and computationally intensive than other pipelines, the trade-off is higher quality alignments and quantification and unless analyzing thousands of samples, is generally not too expensive or time-consuming.

4 Conclusions

We hope that with adoption of this pipeline, the field of high-throughput sequencing can arrive towards a standardized processing protocol for sequencing data and eliminate some of the variability that comes from using a variety of software packages for various steps during read processing. XPRESSpipe will act as a flexible pipeline that will be updated with the best performing packages as future tools are created and benchmarks performed. Additionally, various tools missing from the RNA-seq and ribosome profiling communities have been added as part of this pipeline. With these tools, such as the GTF modification sub-module, genome reference formatting and curation is automated and accessible to the public. Further, by using this pipeline on publicly available data, we highlight XPRESSpipe's utility in being able to re-process publicly available data or personal data to uncover novel biological patterns quickly. Adoption of this tool will aid the average scientist is quickly accessing their data, using the highest-quality methods.

5 Materials and Methods

5.1 Software Dependencies

A list of dependencies required for XPRESSpipe is listed in Table 4. Dependencies for XPRESSplot are listed in Table 5.

Table 4: Summary of dependency software, accession location, and purpose in the XPRESSpipe package.

Package	Purpose	Reference
Python	Primary language	
R	Language used for some statistical modules	
fastp	Read pre-processing	(*13*)
STAR	Reference curation and read alignment	(*12*)
samtools	Alignment file manipulation	(*16*)
bedtools	Alignment file manipulation	(*17*)
deepTools	Alignment file manipulation	(*18*)
Cufflinks	Read quantification (primary)	(*10*)
HTSeq	Read quantification	(*20*)
FastQC	Quality Control	(*23*)
MultiQC	Quality Control	(*42*)
dupRadar	Measure library complexity	(*24*)
pandas	Data manipulation	(*43*)
numpy	Data manipulation	(*29, 30*)
scipy	Data manipulation	(*31*)
sklearn	Data manipulation	(*44*)
matplotlib	Plotting	(*26*)
XPRESSplot	Normalization and matrix manipulation	This paper
rsubread	Dependency for dupRadar	(**?**)
dupRadar	Performs library complexity calculations	(*24*)
deseq2	Performs differential expression analysis	(*32*)

Table 5: Summary of dependency software, accession location, and purpose in the XPRESSplot package.

Package	Purpose	Reference
Python	Primary language	
R	Language used for some statistical modules	
pandas	Data manipulation	(*43*)
numpy	Data manipulation	(*29, 30*)
scipy	Data manipulation	(*31*)
matplotlib	Plotting	(*26*)
seaborn	Plotting	(*27*)
plotly	Plotting	(*45*)
sklearn	Data manipulation	(*44*)
GEOparse	Access GEO data	(*46*)
DESeq2	Perform differential expression analysis	(*32*)
sva	Perform batch correction for known effects with the ComBat function	(*21*)

5.2 GTF Modification

Protein coding genes are identified by the “protein_coding” annotation within `attribute` column of the GTF file. Longest transcripts are determined by calculating the exon space for each transcript associated with a given “gene_id”. If a pre-mature stop is annotated within a transcript, that is considered the end-point of the transcript length. Truncation is performed by identifying the 5’ and 3’ end of each transcript and modifying the given co-ordinates to reflect the given truncation amounts. The amounts to be truncated can be modulated by the user; however, suggested ranges are 45 nt from the 5’ end and 15 nt from the 3’ end, set as the default parameters for the function (*8*). As a given exon may be less than the specified amounts, the function will recursively search exon by exon until the full truncated amount is trimmed.

5.3 Normalization

Equations 1-4 reflect the design of the normalization functions within XPRESSplot.

$$RPM = \frac{(\#\ number\ reads\ per\ gene)\ \cdot\ 1e6}{(\#\ mapped\ reads\ per\ sample)} \tag{1}$$

$$RPKM = \frac{(\#\ number\ reads\ per\ gene)\ \cdot\ 1e6\ \cdot\ 1e3}{((\#\ mapped\ reads\ per\ sample)\ \cdot\ (gene\ length\ (bp))} \tag{2}$$

$$FPKM = \frac{(\#\ number\ fragments\ per\ gene)\ \cdot\ 1e6\ \cdot\ 1e3}{(\#\ mapped\ fragments\ per\ sample)\ \cdot\ (gene\ length\ (bp))} \tag{3}$$

$$TPM = \frac{(\#\ number\ fragments\ per\ gene)\ \cdot\ 1e3\ \cdot\ 1e6}{(gene\ length\ (bp))\ \cdot\ (\#\ mapped\ fragments\ per\ sample)} \tag{4}$$

5.4 Metagene Estimation

Metagene calculations are performed by determining the meta-genomic coordinate *M* for each aligned read, where *L* is the leftmost coordinate of the mapped read and *r* is the length of the mapped read. *S* denotes the start coordinate for the transcript and *l* is the cumulative length of all exons for the given transcript. The subscripted *e* indicates the coordinate is relative to exon space, where intron space is not counting in the coordinate relative to the start of the transcript. Required inputs are an indexed BAM file and an unmodified GTF reference file. For each mapped coordinate, the metagene position is calculated as:

$$M = \frac{|(L_e\ +\ \frac{1}{2}r)\ -\ S|\ \cdot\ 100}{l_e} \tag{5}$$

In the case where a mapped coordinate falls within multiple genes, a penalty is assigned as:

$$c = \frac{1}{n} \tag{6}$$

Where *c* is the count score for a given meta-position and *n* is the number of different transcripts a given coordinate mapped. To be counted or factored into the penalty, the meta-position coordinate must fall within exon space.

5.5 Periodicity

p is the distance from the start coordinate, *L* is the leftmost coordinate of the mapped read, *r* is the length of the mapped read, and *S* denotes the start coordinate for the transcript. The superscript signs associated with *p* indicate strandedness and the subscript *e* indicates the coordinate is relative to exon space. Only reads 28-30 nucleotides long are considered in this analysis.

$$p^{+} = (L_e\ +\ r - 16)\ -\ S \tag{7}$$

$$p^{-} = S\ -\ (L_e\ +\ 16) \tag{8}$$

5.6 rRNA Probe

`rrnaProbe` works on a directory containing fastqc (*23*) zip compressed files to detect over-represented sequences for each sample. These sequences are then collated to create consensus fragments. One caveat is that FASTQC collates on exact matching sequences, but these sequences may be 1 nt steps from each other and a single rRNA probe could be used to effectively pull out all these sequences. In order to handle this situation, XPRESSpipe will combine these near matches. A rank ordered list of over-represented fragments within the appropriate length range to target for depletion is then output. A BLAST (*47*) search on consensus sequences intended for probe useage can then be performed to verify the fragment maps to an rRNA sequence and is thus a suitable depletion probe.

5.7 Confidence Interval Plotting

Confidence intervals within PCA scatterplots generated by XRESSplot are calculated as follows:

1. Compute the covariance of the two principle component arrays, *x* and *y* using the numpy.cov() function.

2. Compute the eigenvalues and normalized eigenvectors of the covariance matrix using the numpy.linalg.eig() function.

3. Compute the θ of the normalized eigenvectors using the numpy.arctan2() function and converting the output from radians to degrees using numpy.deg().

4. Compute the λ of the eigenvalues by taking the square root of the eigenvalues.

5. Plot the confidence intervals over the scatter plot: The center point of the confidence interval is determined from the means of the *x* and *y* arrays. The angle is set equal to θ. The width of the condfidence interval is calculated by

$$w = \lambda_x \cdot ci \cdot 2$$

where *ci* is equal to the corresponding confidence level (i.e. 68% = 1, 95% = 2, 99% = 3). The heighth is similarly computed by

$$h = \lambda_y \cdot ci \cdot 2$$

5.8 Ribosome Profiling Example Data Analysis

Only gene names in common between the original data file and XPRESSpipe output were used for the method comparisons. Genes included in all studies were required to have at least 25 counts across samples to be included in the analysis. Correlations and p-values were calculated using the `scipy.stats.spearman()` function (*48*). Sample count distributions were plotted using Seaborn where density is indicated by width of plot and boxplot designating interquartile range are plotted (*27*). Fold change and translation efficiency plots were created using matplotlib (*26*) and pandas (*43*). Replicates were combined to calculate fold change values and significance between library groups (condition and library type) was calculated using a Benjamini-Hochberg FDR method from `statsmodels.stats.multitest()` (*49*). Gene coverage profiles were generating using IGV (*40*). Figures and analyses can be reproduced using the associated scripts found at https://github.com/XPRESSyourself/manuscript (DOI: XXXXXX).

5.9 Cost Analysis

5.10 Benchmarking

List of abbreviations

UMI - unique molecular identifier, nt - nucleotide,

Ethics approval and consent to participate

Protected TCGA data were obtained through dbGaP project number 21674 and utilized according to the associated policies and guidelines.

Consent for publication

Protected TCGA data were obtained through dbGaP project number 21674 and utilized according to the associated policies and guidelines.

Availability of data and materials

The source code for these packages will be perpetually open source and protected under the GPL-3.0 license. The code can be publicly accessed and installed from https://github.com/XPRESSyourself. Updates to the software are version controlled and maintained on GitHub. Jupyter notebooks and video walkthroughs are included on https://github.com/XPRESSyourself for guiding a user through use of the packages. Documentation is hosted on readthedocs (*50*) at https://xpresspipe.readthedocs.io/en/latest/ and https://xpressplot.readthedocs.io/en/latest/. The publicly available ribosome profiling data are accessible through GEO series accession number GSE65778. TCGA data are accessible through dbGaP accession number phs000178. Code used to create manuscript figures and analyses can be found at https://github.com/XPRESSyourself/manuscript (DOI: XXXXXX).

Competing interests

The authors declare that they have no competing interests.

Funding

J.A.B. received support from the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK) Interdisciplinary Training Grant T32 Program in Computational Approaches to Diabetes and Metabolism Research, 1T32DK11096601 to Wendy W. Chapman and Simon J. Fisher.

Contributions

J.A.B. conceptualized and administered the project; performed all investigation, analysis, visualization, and data curation; provisioned computing resources; acquired funding; amd write the original draft for this study. J.A.B. wrote the software and J.R.B. designed and wrote the rRNA Probe sub-module. J.A.B., J.T.M., A.J.B., and Y.O. performed software and documentation validation. J.A.B., J.R.B., M.T.H., and J.G. and developed the methodology. J.P.R, M.T.H., J.G., and A.R.Q. supervised the study. All authors were involved in reviewing and editing the manuscript.

Acknowledgments

The authors wish to thank Mark Wadsworth, Ryan Miller, and Michael Cormier for helpful discussions on pipeline design. They also wish to thank Cameron Waller for helpful discussions related to pipeline design and analysis. The support and resources from the Center for High Performance Computing at the University of Utah are gratefully acknowledged. The computational resources used were partially funded by the NIH Shared Instrumentation Grant 1S10OD021644-01A1.

References

1. S. Byron, K. V. Keuren-Jensen, D. Engelthaler, J. Carpten, D. Craig, *Nat Rev Genet* **17**, 392–393 (2016).

2. N. Ingolia, S. Ghaemmaghami, J. Newman, J. Weissman, *Science* **324**, 218 (2009).

3. Z. Costello, H. Martin, *NPJ Syst Biol Appl* **4** (2018).

4. V. Funari, S. Canosa, *Science* **344**, 653 (2014).

5. M. Gerashchenko, V. Gladyshev, *Nucleic Acids Res* **42** (2014).

6. C. Artieri, H. Fraser, *Genome Res* **24**, 2011 (2014).

7. J. Hussmann, S. Patchett, A. Johnson, S. Sawyer, W. Press, *PLoS Genet* **11** (2015).

8. N. McGlincy, N. Ingolia, *Methods* **126**, 112 (2017).

9. D. Weinberg, *et al.*, *Cell Rep* **14**, 1787 (2016).

10. C. Trapnell, *et al.*, *Nat Protoc* **7** (2012).

11. Anaconda, Anaconda software distribution, `https://anaconda.com`.

12. A. Dobin, *et al.*, *Bioinformatics* **29**, 15 (2013).

13. S. Chen, Y. Zhou, Y. Chen, J. Gu, *Bioinformatics* **34** (2018).

14. Y. Fu, P. Wu, T. Beane, P. Zamore, Z. Weng, *BMC Genomics* **19** (2018).

15. G. Baruzzo, *et al.*, *Nat Methods* **14**, 135–139 (2017).

16. H. Li, *et al.*, *Bioinformatics* **25**, 2078 (2009).

17. A. Quinlan, I. Hall, *Bioinformatics* **26**, 841 (2010).

18. F. Ramírez, F. Dündar, S. Diehl, B. Grüning, T. Manke, *Nucleic Acids Res* **42** (2014).

19. C. Robert, M. Watson, *Genome Biol* **16** (2015).

20. S. Anders, P. Pyl, W. Huber, *Bioinformatics* **31**, 166 (2015).

21. J. Leek, W. Johnson, H. Parker, A. Jaffe, J. Storey, *Bioinformatics* **28** (2012).

22. C. Evans, J. Hardin, D. Stoebel, *Brief Bioinform* **19**, 776–792 (2018).

23. S. Andrews, Fastqc, `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/` (2010).

24. S. Sayols, D. Scherzinger, H. Klein, *BMC Bioinformatics* **17**, 428 (2016).

25. F. Lauria, *et al.*, *PLoS Comput Biol* **14** (2018).

26. J. Hunter, *Computing In Science & Engineering* **9**, 90 (2007).

27. M. Waskom, et al (2012).

28. F. Pedregosa, *et al.*, *Journal of Machine Learning Research* **12**, 2825 (2011).

29. T. Oliphant, *A guide to NumPy* (Trelgol Publishing, USA, 2006).

30. S. van der Walt, S. Colbert, G. Varoquaux, *Computing in Science Engineering* **13**, 22 (2011).

31. E. Jones, T. Oliphant, P. Peterson, *et al.*, Scipy: Open source scientific tools for python, `http://www.scipy.org/` (2001).

32. M. Love, W. Huber, S. Anders, *Genome Biol* **15** (2014).

33. D. Santos-Ribeiro, L. Godinas, C. Pilette, F. Perros, *Drug Discov Today* **23** (2018).

34. H. Rabouw, *et al.*, *Proc Natl Acad Sci U S A* **116** (2019).

35. J. Tsai, *et al.*, *Science* **359** (2018).

36. C. Sidrauski1, A. McGeachy, N. Ingolia, P. Walter, *eLIFE* (2015).

37. A. Choua, *et al.*, *Proc Natl Acad Sci U S A* **114** (2017).

38. D. Kim, *et al.*, *Genome Biol* **14** (2013).

39. G. Baruzzo, *et al.*, *Nat Methods* **14**, 135 (2017).

40. J. Robinson, *et al.*, *Nat Biotechnol* **29**, 24 (2011).

41. R. Tunney, *et al.*, *Nat Struct Mol Biol* **25**, 577 (2018).

42. P. Ewels, M. Magnusson, S. Lundin, M. Käller, *Bioinformatics* **32**, 3047–3048 (2016).

43. W. McKinney, *Proc of the 9th Python in Science Conf* pp. 51–56 (2010).

44. .

45. .

46. .

47. S. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman, *J Mol Biol.* **215**, 403 (1990).

48. F. Liesecke, *et al.*, *Sci Rep* **8** (2018).

49. S. Seabold, J. Perktold, *9th Python in Science Conference* (2010).

50. Read the docs, `https://readthedocs.org/`.