# XPRESSyourself: Automating and Strengthening the High-Throughput Sequencing Toolkit

Jordan A. Berg,[1] Jeffrey T. Morgan,[1] Jonathan R. Belyeu,[2] Alex J. Bott,[1]
Yeyun Ouyang,[1] Jason Gertz,[3] Aaron R. Quinlan,[2,4,5] 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.

**Nucleic acid sequencing is a routine and very powerful tool in biological and clinical research. However, for average users, computational bottlenecks often exist. XPRESSyourself is a ribosome profiling and RNA-seq analytical pipeline that aims to eliminate these barriers, standardize *in silico* protocols, and decrease time-to-discovery from data. Additionally, XPRESSyourself introduces and updates tools missing from the current ribosome profiling and RNA-seq computational toolkits. Using XPRESSyourself, we were able to identify in a matter of hours putative mechanisms behind neurodegeneration during acute stress using publicly available ribosome profiling data, highlighting its utility to rapidly uncover novel biological insight.**

## Keywords

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

## 1 Background

High-throughput sequencing data has revolutionized biomedical, industrial, and basic science research. Specifically, RNA-seq has proven to be the forerunner technology for high quality RNA quantification within the last two decades. RNA-seq involves isolating the RNA fragments from a population of cells, incorporating these fragments into cDNA libraries, and assembling the sequenced reads to a reference genome or transcriptome to measure relative transcript abundance, differential splice variants, sequence polymorphisms, and more (*1*). High-throughput sequencing technologies have been developed for a variety of technologies such as DNA sequencing, ChIP-seq, single-cell RNA-seq, and more recently, ribosome profiling (*2*).

While vast strides have been made to implement and perfect 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 intricacies of the different tools used in processing RNA-seq data can be challenging and problematic. Moreover, many users are not aware of the most appropriate or most up-to-date tools or settings for their application (*3,4*). Even for the experienced user, developing robust, automated pipelines that accurately process and assess quality of these datasets can be laborious. The variability that inevitably arises with each lab or bioinformatics core designing and using their own pipelines is also a significant challenge in the field.

While RNA-seq is a fairly matured technology, there are still an abundance of biases and idiosyncrasies associated with each analytical method or tool, of which a beginner user may not be aware. Additionally, few if any pre-existing pipelines or toolkits 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 automatically facilitate this important step (*8, 9*).

Several computational pipelines for sequencing have emerged that intend to tackle various aspects of these bottlenecks, but many suffer from usability issues, are not easily modifiable, or sacrifice quality for speed. For example, a simple internet search for RNA-seq pipelines will reveal several classes of pipeline. The first class is a tutorial titled a pipeline. Many instances of these can be found (*10, 11*); however, they are not automated and are often outdated. The second class is an automated pipeline, but requires extensive manual configuration (*12–15*). The third class is an automated pipeline, but requires programmatic configuration to modify common parameters (*16, 17*). Perhaps, the most user-friendly example is Galaxy, but in cases like its 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.

In response to these issues surrounding the automation of sequencing technology, we designed the XPRESSyourself bioinformatics suite for processing and analyzing high-throughput expression data. Architecturally, this suite is designed to work fast, while not sacrificing quality for speed. Each step of the pipeline utilizes the state of the art software package for that task, having been previously vetted by peer-reviewed benchmarking studies. Additionally, the scaffold that underlies the pipeline structure is designed such that updating and testing of a new module is facile for a trained bioinformatician, thus continuously the best options available to the entire community, regardless of expertise.

With the XPRESSpipe package within XPRESSyourself, 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 characteristic 1 codon step of the ribosome. These measurements are particularly helpful for ribosome profiling experiments due to the unique characteristics of the ribosome footprint-sized libraries (usually around 21-30 nt). XPRESSpipe also includes a metagene analysis sub-module that shows the distribution of the relative position of all aligned reads across a representative transcript to ensure that no 5' or 3' biases occured during library preparation. As PCR duplicates can arise during library preparation, a library complexity visualization sub-module is included in the pipeline to ensure that PCR duplicates are minimized in the library and a robust population of different transcripts were captures during the library preparation. Additionally, the XPRESSplot package within XPRESSyourself provides tools to perform the bulk of sequence analysis and generation of figures for publication, where many plot generation protocols that frequently require several hundred lines of code are condensed to a single line with minimal input from the user. XPRESSyourself suite tools are perpetually open source under a GPL-3.0 license at https://github.com/XPRESSyourself.

# 2 Results

## 2.1 XPRESSpipe

XPRESSpipe contains automated pipelines for ribosome profiling, single-end, and paired-end RNA-seq. 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 that will be continually updated as features are updated or changed (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 |
| `-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: HTSeq-count (*18*)) |
| `--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
# Dependencies can be downloaded as follows:
$ conda env create -v -n xpresspipe -f requirements.yml
$ source activate xpresspipe # This will be executed each time you log back into the command line
# XPRESSpipe is installed as follows:
$ 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 user's system for Anaconda (*19*) and install

required dependencies. If working on a compute cluster without administrative privileges, Anaconda should be installed manually first before installing XPRESSpipe and associated dependencies. Anaconda and XPRESSpipe should be directed to the user's `.local` directory, as follows:

Listing 2: Anaconda installation on compute node.

```
# Clean environment of pre-installed dependencies by admins
$ cd $HOME
$ module purge

# Get the latest anaconda2 or anaconda3 distribution and make executable
$ curl -O https://repo.anaconda.com/archive/Anaconda3-5.3.0-Linux-x86_64.sh
$ chmod 700 Anaconda3-5.3.0-Linux-x86_64.sh

# Install Anaconda in batch mode (assumes license agreed upon, setup prefix directory, and skip pre-
    and post- install scripts)
export INSTALL_DIR=$HOME/softwares/anaconda3/5.3.0
./Anaconda3-5.3.0-Linux-x86_64.sh -b -p $INSTALL_DIR -s
```

Listing 3: Source installation on compute node.

```
# Download and unzip archived version from https://github.com/XPRESSyourself/XPRESSpipe/releases
$ cd XPRESSpipe
# Dependencies can be downloaded as follows:
$ conda env create -v -n xpresspipe -f requirements.yml # These may need to be installed manually to
    the user's profile depending on compute node setup
$ source activate xpresspipe # This will be executed each time you log back into the command line
# XPRESSpipe is installed as follows:
$ python setup.py install --prefix ~/.local/bin
```

Docker images come with these dependencies pre-installed and can be accessed as follows:

Listing 4: 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 `.fq`, `.fastq`, or `.txt`
- Paired-end reads should end in `.read1/2.fq` or `.r1/2.fq`, where `.fq` could also be `.fastq` or `.txt`
- Read files can be compressed as `.zip` or `.gz`. Decompression will be handled automatically by XPRESSpipe
- The base 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 to which the alignment software will map reads. For the purposes of the current version of XPRESSpipe, a STAR (*20*) 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 in the next section. Additionally, any chromosome `.fasta` files should be

placed 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. This entire process is handled with the `curateReference` sub-module for ease of use to the user.

Listing 5: curateReference example

```
$ xpresspipe curateReference -o /path/to/output/location/ \
                             -f /path/to/fasta/genome/ \
                             -g /path/transcripts.gtf \
                             --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. By providing the `--protein_coding` argument, only protein-coding genes are retained in the GTF file, which acts as a masking of any reads aligning to non-coding regions of the genome. In most eukaryotes, mRNAs undergo alternative splicing. However, some quantification tools will consider the multiple 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 Ensembl canonical transcrtipt (*21*) is retained for each gene in the GTF file. However, if using Cufflinks to quantify reads (discussed further in the next section), this is may not be recommended as Cufflinks is optimized to quantify isoform abundances (*22*).

For the purposes of ribosome profiling, where 5' and 3' transcript biases are frequent (*8, 9*), the 5' and 3' ends of each transcript's coding space 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. These values are set to defaults of 45 nt for 5' truncation and 15 nt for 3' truncation, as is common in the field (*8*), but these can be modified using the `--truncate_5prime` or `--truncate_5prime` arguments.

#### 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 (*23*). 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 (*24*).

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 (*20,25*). XPRESSpipe is capable of performing 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 when the user specifies to use deduplicated alignments for downstream processing. Use of de-duplicated alignment files may be advisable in situations where the library complexity profiles (discussed below) exhibit high duplication levels; however, files with duplicates are used for downstream processing by default. These steps are performed using samtools (*26*). Optionally, BED and bigWig files can also be output. These conversions are handled by bedtools (*27*) and deeptools (*28*).

4. **Read Quantification**: XPRESSpipe quantifies read alignments for each input file using Cufflinks (*22,29*). Use of Cufflinks will output a read table that has been normalized already as well as a counts table for downstream applications where required. If masking of non-coding RNAs is desired, a `protein_coding` modified GTF file should be provided for the `--gtf` argument. Optionally, the user can use HTSeq with the `intersection-nonempty` method as default (*18,29*). Our rationale for this decision is that it conforms closer to TCGA standards. Additionally, most users will be using longest-transcript only GTF so will be less concerned with isoform abundance estimates. However, there are limitations of HTSeq, therefore Cufflinks is recommended.

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. Current 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 (*30*). 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 by way of XPRESSplot by the ComBat package (*31*).

6. **Quality Control**: An important step in RNA-seq analysis is proper quality control of sequencing samples to ensure the interpreted downstream 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 (*32*) after trimming. By assessing the read distribution of each sample, the user can ensure the expected read size was sequenced. This is particularly helpful in ribosome profiling experiments for verifying the requisite 21-30 nt ribosome footprints were inserted into the sequencing library successfully (*8*). Metrics are compiled and plotted by XPRESSpipe.

- **Library Complexity**: Analyzing library complexity is an effective method for analyzing the robustness of a sequencing experiment in capturing various, unique 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. This analysis is performed using dupRadar (*33*) where inputs are the PCR duplicate-tagged BAM files output by XPRESSpipe. Duplicate tagging is performed by samtools (*26*). Metrics are then compiled and plotted by XPRESSpipe.
- **Metagene Estimation Profile**: In order to identify any general 5' or 3' transcript biases in captured reads, a metagene profile can be created for each sample. This is performed by determining the metagenomic 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 (*34*). The same inputs are required as for the `metagene` submodule. This method is 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, sequencing of these reads becomes highly repetative and often biologically uninteresting in the context of transcription. 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 for 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 analyze the over-represented sequences between footprint libraries after adaptor and quality trimming and compile conserved k-mers across 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` in the command line. Figure 1 provides an example of the output file scheme for XPRESSpipe. For

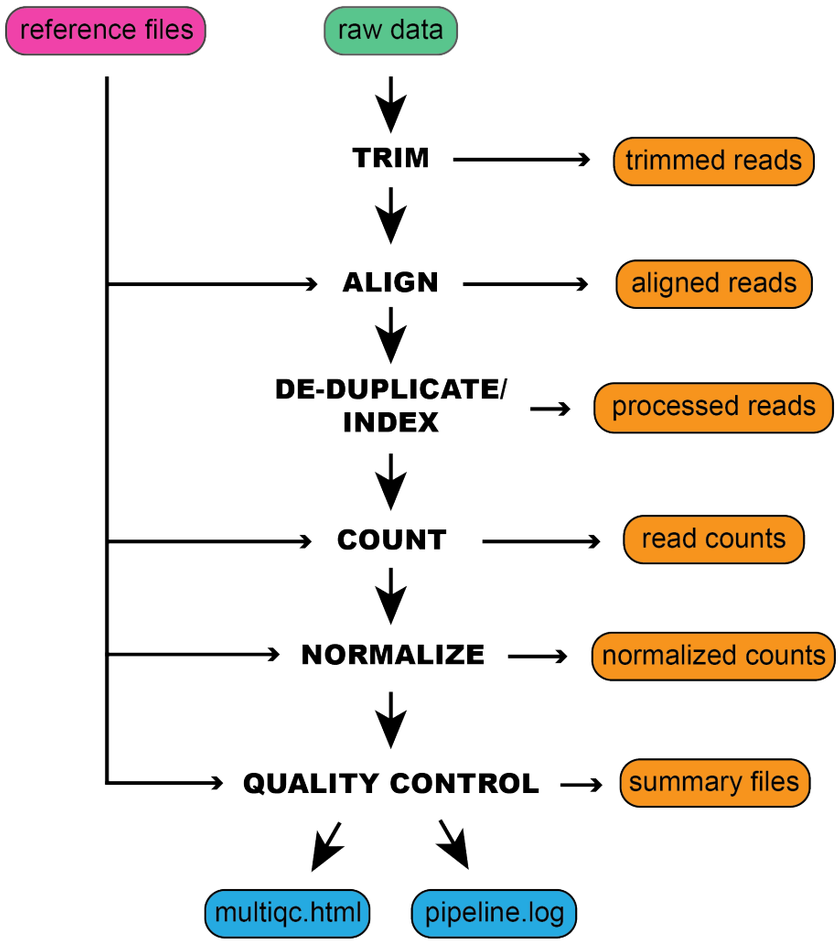


Figure 1: An example schematic of the inputs required by XPRESSpipe and organization of the ouputs.

a complete pipeline run, the user can expect BAM alignment files, a collated count table of all samples in the experiment, and quality control figures and metrics. 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 Pythonic library of analysis and plotting tools that builds upon existing packages, such as Matplotlib (*35*) and Seaborn (*36*) to generate flexible, specific analyses and plots frequently used by biological researchers that can each be executed in a single line of code rather than tens to hundreds. Additionally, many included features are currently available in an R or other programming language package but not in a Python package. Brief summaries of key components of this package, as well as descriptions 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 detailed instructions for other features currently in the toolkit, as well as for future features to be added. While XPRESSplot is designed for handling transcriptomics datasets, is also capable in many cases of handling other omics datasets, such as proteomics or metabolomics.

### 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 (*30*). Other normalizations, such as mean centering of *j* features (i.e. genes or other items) by sklearn's preprocessing module (*37*). 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 that are particularly useful and refer the reader to the documentation for further details and examples of additional analysis features.

- **Principle Components Analysis**: Principle components analysis (PCA) for the data matrix is computed using Python's scikit-learn package (*37*) and desired principle components are plotted in a scatter plot via the matplotlib (*35*) and seaborn (*36*) 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 labels. 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 (*38, 39*), a feature lacking from Pythonic PCA packages.

- **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 (*40*). 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 (*41*).

## 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 one ribosome profiling dataset and a subset of TCGA samples to showcase the utility of XPRESSpipe for rapidly extracting potentially 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 in response 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 activation of the ISR (*42*). While acute ISR activation is essential for proper cell survival, long periods of sustained ISR activity can be damaging. A recently discovered small molecule inhibitor of the ISR, ISRIB, has demonstrated therapeutic potential and relative lack of side-effects. Interestingly, ISRIB is able to suppress a chronic low-grade ISR, while it does not inhibit an acute, high-grade ISR. It has also been shown to be neuroprotective in mouse models of acute neurological damage (*43–46*).

UPDATE: ADD MORE EXAMPLES HERE –¿ JR: It is my impression that there are many, many examples of ISRIB efficacy demonstration in animals. We should try to be as comprehensive as possible.

A recent study (GSE65778) utilized ribosome profiling in order to better define the mechanisms of ISRIB action on the ISR, modeled by tunicamycin (Tm) treatment of HEK293T cells (*45*). Some key findings from this study were that during acute ISR, a specific subset of canonical stress-related transcription factors mRNAs were translationally enhanced during ISR and returned to levels seen in untreated cells when ISR-induced cells were additionally treated with ISRIB. In order to showcase the utility of XPRESSpipe in re-analyzing ribosome profiling and sequencing datasets, we re-processed and analyzed this dataset using the more current *in silico* techniques included in the XPRESSpipe package to shed more light on the translational mechanisms of the ISR and ISRIB. Compared to the raw count data made available in the original manuscript, samples showed comparable alignment rates between the two analytical regimes (Spearman R values $>$0.90) (Figure 2A). This is in spite of the fact that the methods section of the original publication outlines the use of now outdated software, such as TopHat2 (*47*), which has a documented higher false positive alignment rate compared to the current state-of-the-art tool, STAR (*20, 48*). This bias is reflected in the Spearman correlations in Figure 2A. Processing of replicates within the XPRESSpipe method showed excellent correlation (Spearman R values all $>$0.98) (Figure 2B). Interestingly, de-duplication of ribosome footprint samples appeared to improve the count values correlations, which may be in part due to the inherent bias of certain sequences by CircLigase, used in the creation of these libraries (*45, 49*).

XPRESSpipe-processed samples discovered similar canonical targets of translational regulation during ISR compared to the original study, such as ATF4, ATF5, and PPP1R15A (Figure 3A, highlighted in magenta) (*45*). Of note, the fold-change in ribosome occupancy of ATF4 (6.75) from XPRESSpipe-processed samples closely matched the estimate from the original publication (6.44). Other targets highlighted in the original study (*45*), such as ATF5, PPP1R15A, and DDIT3 also increased in their ribosome occupancy fold-changes (5.83, 2.49, and 3.29; respectively) (Figures 3A, B.

In the original study, the genes translationally down-regulated genes were identified. However, re-analyzing these data with the updated XPRESSpipe methodology identifies some genes that may play a role in the neurodegenerative effects of ISR and the neuroprotective properties of ISRIB. For example, one would expect putative targets to be significantly translationally down-regulated during ISR and that these same targets would show no change in the ISR + ISRIB condition. Using this paradigm as a guide, we identified seven genes following this pattern of translational regulation, as listed in Table 2 (descriptions sourced from https://www.genecards.org/,

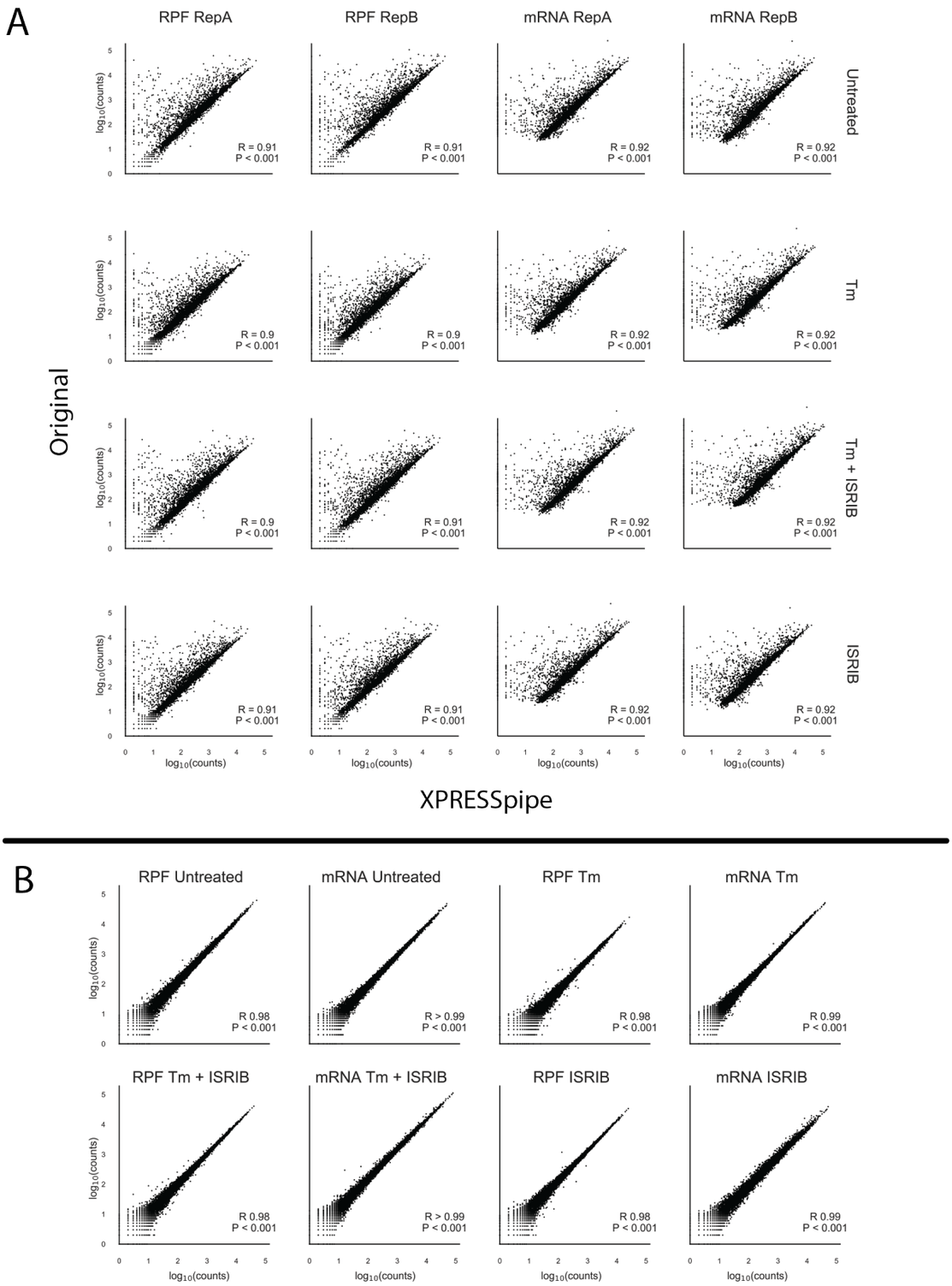


Figure 2: Broad sample quality control and comparision to original processing. A) Cross-processing comparisions between original manuscript and XPRESSpipe. B) Intra-processing comparisions between replicates for XPRESSpipe processing. Note: All R values reported are Spearman R values. All axes are $\log_{10}$(counts).

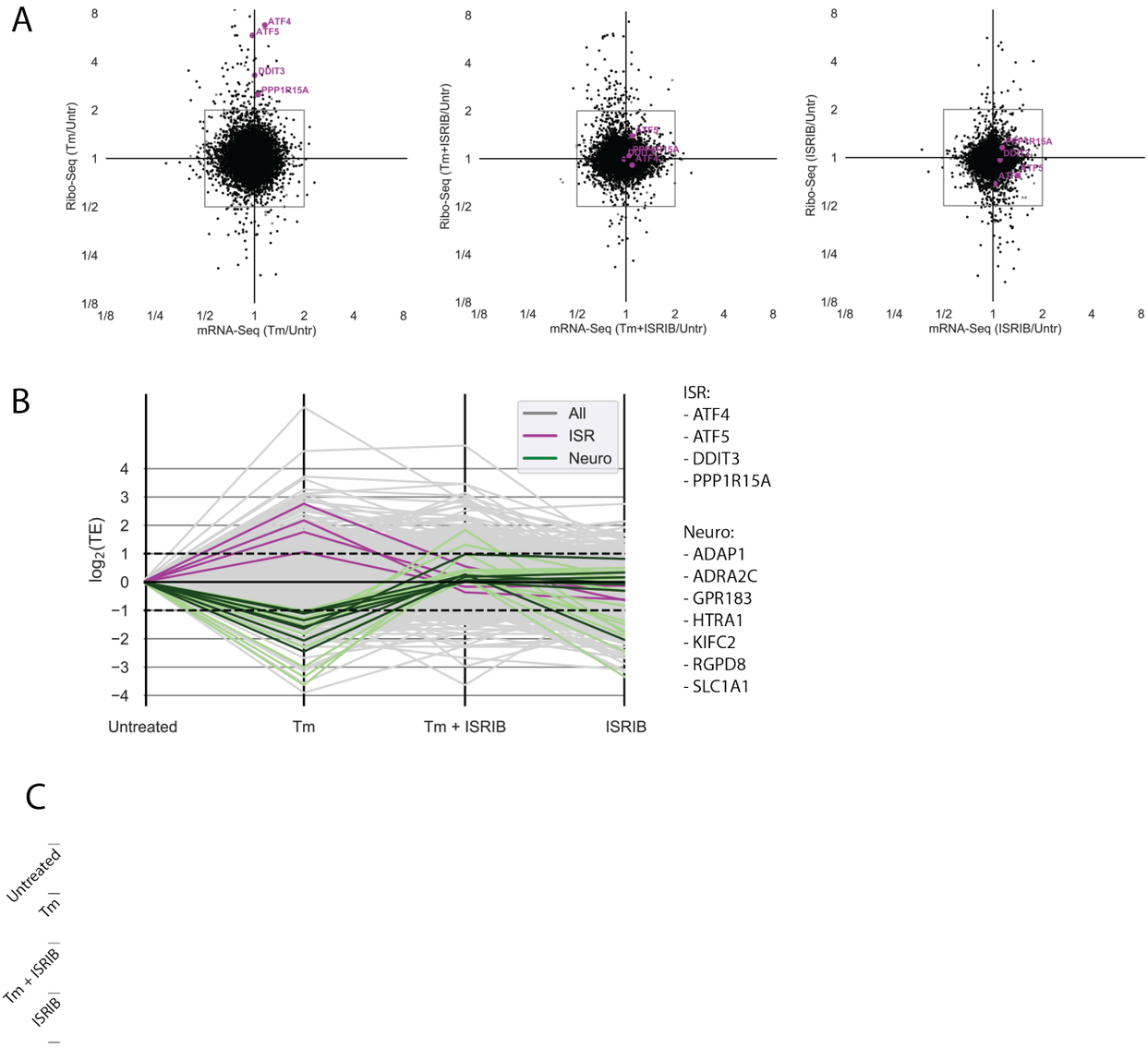


Figure 3: Biological validation and insight into previously published ISR model ribosome profiling data. A) Fold change for each drug condition compared to untreated for the ribosome profiling and RNA-seq data. ISR canonical targets highlighted in original study are in magenta. B) Changes in $\log_2$ Translation Efficiency compared to Untreated for each drug condition for each of the translationally down-regulated genes passing previous discussed thresholding paradigms. Grey lines are all genes, magenta lines are ISR canonical targets from original study, light green lines are all genes fitting thresholding, and dark green lines are neurologically-annotated genes within the thresholding limits. C) Read coverage plot for UPDATE for all ribosome profiling samples show read coverage across the transcript.

https://www.ncbi.nlm.nih.gov/gene/, and https://www.uniprot.org/uniprot/; annotations accessed 19 Jun 2019) 3B. These targets act as interesting putative targets for further follow-up in a model better mirroring the neurological environment than HEK-293T cells. Further analysis of these potential hits is strengthened by investigating read pile-ups along these genes in IGV (*50*), where footprint coverage is observed across the transcript for all ribosome profiling samples except the Tm-treated samples (Figures 3C. This provides further support to this observation as use of CircLigase in the library preparation can bias certain reads' incorporation in sequencing libraries, whereas there appears to be none in these samples (*49*).

Table 2: Translationally down-regulated genes during acute Tm treatment and normal regulation during Tm + ISRIB treatment.

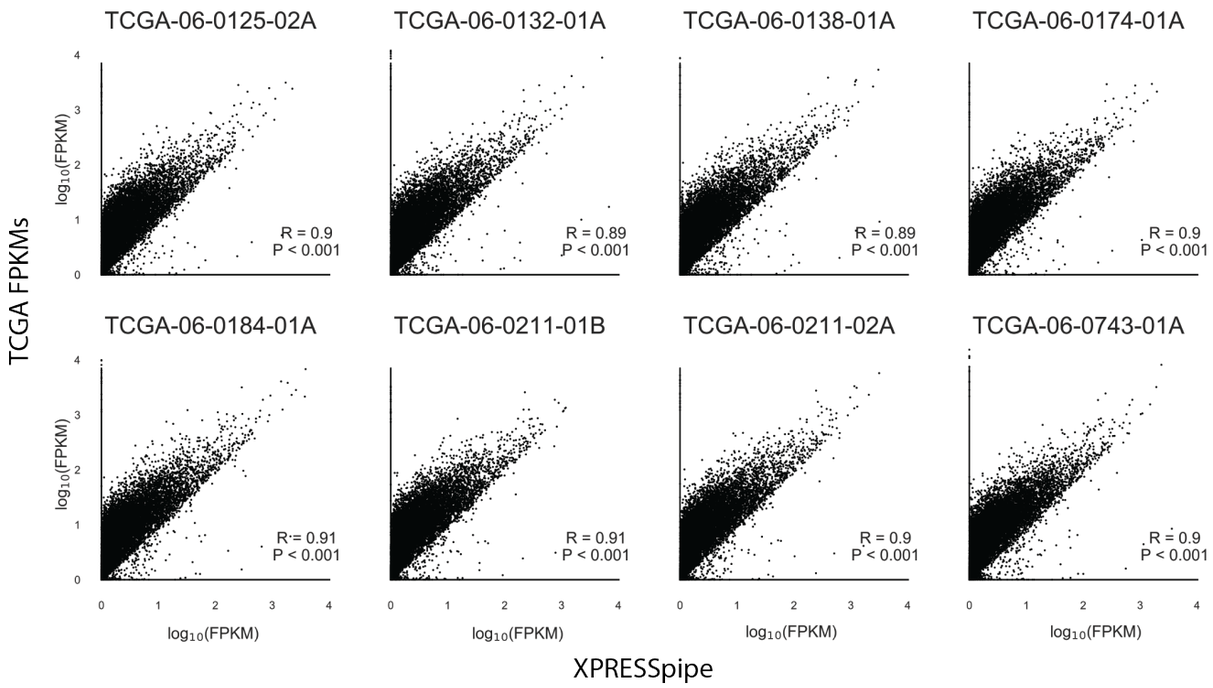


Figure 4: Pipeline validation using publicly available TCGA FPKM data. Correlations were calculated between publicly available FPKM counts from TCGA samples and the calculated FPKM values using XPRESSpipe on the raw sequence data. Note: All R values reported are Spearman R values. All axes are $\log_{10}$(counts).

| Gene Name | Relevant Description |
|---|---|
| ADAP1 | Overexpressed in fetal brain, frontal cortex, and spinal cord. Expressed at highest levels in brain. Involved in cell signaling events. |
| ADRA2C | Critical role in neurotransmitter release from sympathetic nerves and adrenergic neurons. Role in positive regulation of neuron differentiation. |
| GPR183 | Chemotactic receptor in dendritic cells. Expression required for splenic dendritic cell homeostasis, localization, and induction of B- and T-cell responses. Regulates astrocyte migration. |
| HTRA1 | Moderate overexpression in brain. Suggested role in cell growth regulation. Variations in promoter region increase susceptability to age-related macular degeneration type 7. Associated with other neurodegenerative diseases. May regulate neuronal survival and maturation during development. |
| KIFC2 | Overexpressed in cerebellum. May function as motor for multivesicular body-like organelles in dendrites (by similarity). |
| RGPD8 | Moderate expression in brain. Embryonic expression in neural tube. Associated behavioral/neurological phenotypes. |
| SLC1A1 | Dense expression in substantia nigra, red nucleus, hippocampus, and cerebral cortical layers. Member of high-affinity glutamate transporter. In brain, crucial for terminating postsynaptic action of the neurotransmitter glutamate. Responsible for maintaining glutamate concentrations below neurotoxic levels. |

### 2.3.2 Performance Validation Using TCGA Data

To validate the general design and reliability of the XPRESSpipe pipeline, we used raw TCGA sequence data, processed it using XPRESSpipe, and compared the output FPKM values to those publicly available through TCGA. Spearman R values were >UPDATE, indicating integrity of pipeline design.

UPDATE add methods

### 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 Web Services (AWS) (https://aws.amazon.com/) can be used to process sequencing data using XPRESSpipe. For example, the ribosome profiling dataset of 32 raw sequence files 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 3.

UPDATE

Table 3: XPRESSpipe processing statistics for dataset GSE65778.

| **Metric** | **Value** |
|---|---|
| Elapsed Real Time | 07h39m47s |
| Total CPU Time | 1d04h34m36s |
| Allocated CPUs | 16 |
| Allocated Memory per Node | 62.50GB |
| Maximum RAM of all tasks | 62.79GB |

Based on these metrics, if HPC services weren't locally available for a user, one could use Amazon Web Services to process the data for relatively little money. For a comparable run, storage cost would amount to around 14 USD/month on Amazon S3 storage and compute cost for a similar computational node for the given elapsed time would cost around 6 USD using Amazon EC2 On-Demand (however, significantly reduced rates are available if using Spot instances).

# 3 Discussion

We have described herein a new software suite, XPRESSyourself, a collection of tools 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 or 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 through continuous curation by XPRESSyourself team members to ensure the pipeline is utilizing 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 problems or technical biases that may be present in their samples.

An additional problem XPRESSpipe addresses is the incorrect use of these software tools, which is especially important for those coming from a non-computational background. 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). These users can also be assured they are using the most up-to-date standard for RNA-seq processing and analysis.

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 CDS space and rRNA probe design aids for removing contaminating rRNA sequences in ribosome profiling libraries that are difficult to remove with commercial kits.

We demonstrated 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 errors. 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 rapidly process and analyze their data, save money, and put missing or incomplete computational tools required for ribosome profiling and RNA-seq into the hands of the average user.

While admittedly the XPRESSyourself pipeline may be somewhat 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

With the adoption of this pipeline, the field of high-throughput sequencing, particularly within ribosome profiling, can continue to standardize the 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 ribosome profiling and RNA-seq 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 | (*23*) |
| STAR | Reference curation and read alignment | (*20*) |
| samtools | Alignment file manipulation | (*26*) |
| bedtools | Alignment file manipulation | (*27*) |
| deepTools | Alignment file manipulation | (*28*) |
| Cufflinks | Read quantification (primary) | (*22*) |
| HTSeq | Read quantification | (*18*) |
| FastQC | Quality Control | (*32*) |
| MultiQC | Quality Control | (*51*) |
| dupRadar | Measure library complexity | (*33*) |
| pandas | Data manipulation | (*52*) |
| numpy | Data manipulation | (*38, 39*) |
| scipy | Data manipulation | (*40*) |
| sklearn | Data manipulation | (*53*) |
| matplotlib | Plotting | (*35*) |
| XPRESSplot | Normalization and matrix manipulation | This paper |
| rsubread | Dependency for dupRadar | |
| dupRadar | Perform library complexity calculations | (*33*) |
| deseq2 | Perform differential expression analysis | (*41*) |

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 | (*52*) |
| numpy | Data manipulation | (*38, 39*) |
| scipy | Data manipulation | (*40*) |
| matplotlib | Plotting | (*35*) |
| seaborn | Plotting | (*36*) |
| plotly | Plotting | (*54*) |
| sklearn | Data manipulation | (*53*) |
| GEOparse | Access GEO data | (*55*) |
| DESeq2 | Perform differential expression analysis | (*41*) |
| sva | Perform batch correction for known effects with the ComBat function | (*31*) |

### 5.2 GTF Modification

Protein coding genes are identified by the "protein_coding" annotation within `attribute` column of the GTF file. UPDATE 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 coordinates 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 CDS may be less than the specified amounts, the function will recursively search CDS by CDS 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 Quality Control Summary Plotting

Summary plots are created using pandas (*52*) and matplotlib (*35*). Kernel density plots for library complexity analyses are created using numpy (*38, 39*) and scipy's `gaussian_kde` function (*40*).

### 5.5 Metagene Estimation

Metagene calculations are performed by determining the meta-genomic coordinate *M* for each aligned read, where $L_e$ 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 $l_e$ 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.

### 5.6 Periodicity

$p$ is the distance from the start coordinate, $L_e$ 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. The penalty is calculated in the same manner as in the `metagene` sub-module.

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

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

### 5.7 rRNA Probe

`rrnaProbe` works on a directory containing fastqc (*32*) 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 (*56*) 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.

### 5.8 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 $\theta$ of the normalized eigenvectors using the numpy.arctan2() function and converting the output from radians to degrees using numpy.deg().
4. Compute the $\lambda$ 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 $\theta$. 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

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

### 5.9 Ribosome Profiling Example Data Analysis

The following xpresspipe command was run to process the raw data, available from GEO. Reference files were taken from Ensembl Human build GRCh38.p12.

Listing 6: Ribosome profiling processing command.

```
# Generate reference files
$ xpresspipe curateReference -o $SCRDIR/references/human_reference -f
    $SCRDIR/references/human_reference -g $SCRDIR/references/human_reference/transcripts.gtf
    --sjdbOverhang 49 -l -p -t
# Execute pipeline
$ xpresspipe riboseq -i $SCRDIR/input -o $SCRDIR/output -r $REF --gtf $REF/transcripts_LCT.gtf -e
    isrib_riboprof -a CTGTAGGCACCATCAAT --method RPKM --sjdbOverhang 49
```

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 (*57*). Sample count distributions were plotted using Seaborn where density is indicated by width of plot and boxplot designating interquartile range are plotted (*36*). Fold change and translation efficiency plots were created using matplotlib (*35*) and pandas (*52*). 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()` (*58*). Gene coverage profiles were generating using IGV (*50*). Figures and analyses can be reproduced using the associated scripts found at https://github.com/XPRESSyourself/manuscript (DOI: XXXXXX).

Translation efficiency was calculated as follows:

$$\Delta TE = \frac{RPF_{treatment}}{mRNA_{treatment}} - \frac{RPF_{untreated}}{mRNA_{untreated}} \tag{9}$$

### 5.10 Cost Analysis

Cost analysis was performed by accessing run logs from the HPC and using published AWS prices (https://aws.amazon.com/ec2/pricing/on-demand/, https://aws.amazon.com/s3/pricing/, accessed 28 May 2019) to calculate relative cost for a similar run.

## 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 (*59*) 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. `https://www.encodeproject.org/rna-seq/`.

11. `https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/`.

12. `https://github.com/PavlidisLab/rnaseq-pipeline`.

13. `https://github.com/nf-core/rnaseq`.

14. `https://github.com/UMCUGenetics/RNASeq`.

15. `https://github.com/cellgeni/rnaseq`.

16. `https://github.com/dnanexus/tophat_cufflinks_rnaseq`.

17. `https://www.nextflow.io/example4.html`.

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

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

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

21. `https://uswest.ensembl.org/Help/Glossary`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

53. L. Buitinck, *et al.*, *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), pp. 108–122.

54. P. T. Inc., Collaborative data science (2015).

55. R. Gumienny, Geoparse, `https://github.com/guma44/GEOparse`.

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

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

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

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