XPRESSyourself: Automating and Democratizing High-Throughput Sequencing

With the advent of high-throughput sequencing platforms, expression profiling is becoming common-place in medical research. However, for the general user, a computational overhead often exists. The XPRESSyourself suite aims to reduce these barriers to entry for those with limited or advanced computational experience alike and create a series of tools aimed at standardizing and increasing throughput of data processing and analysis. The XPRESSyourself suite is currently broken down into two software packages. The first, XPRESSpipe, automates the pre-processing, alignment, quantification, normalization, and quality control of single-end, paired-end RNAseq, and ribosome profiling sequence data. The second, XPRESStools, is a Python toolkit for expression data analysis, compatible with private or RNAseq datasets. This software suite is designed to be easily be modified, and additional packages can be included for processing of other data types in the future, such as CHIPseq or genome alignment. In addition, this package offers several new tools useful in processing RNA-seq data, specifically for ribosome profiling. We validated the performance of this suite by processing and analyzing pubilcly available datasets and comparing the output with published results. Use of this pipeline will close gaps in the stardardization of RNA-seq processing and democratize the RNA-seq bioinformatic experience.

XPRESSyourself is freely available on GitHub: https://github.com/XPRESSyourself

1 Introduction

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 also been adopted to create technologies such as single-cell RNA-seq, capable of assaying the transcriptional profile cell by cell; and ribosome profiling, which measures ribosome occupancy and translation efficiency (*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 process and check datasets can be laborious and introduce variability to data processing.

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. Additionally, few if any 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 aid the general user in doing so (*8, 9*).

In response to the issues surrounding the automation and democratization of sequencing technology, we created the XPRESSyourself bioinformatics suite for processing and analyzing high-throughput expression data. In creating this tool, we focused on usefulness, usability, reliability, efficiency, and flexibility (*10*). 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 RNA-seq and ribosome profiling 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 measures are particular helpful for ribosome profiling experiments. 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 XPRESStools package, tools are provided to perform the bulk of sequence analysis and generation of figures for publication.

2 Materials and Methods

2.1 XPRESSpipe

XPRESSpipe pipelines for single-end RNA-seq, paired-end RNA-seq, and ribosome profiling offer a handful of tunable parameters to the user, while keeping most parameters hidden to maintain 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 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	
`-a, --adaptors`	Specify adaptor as string (only one allowed) – 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)
`--output_bed`	Include option to output BED files for each aligned file
`--output_bigwig`	Include flag to output bigwig files for each aligned file
`--method`	Normalization method to perform (options: "RPM", "TPM", "RPKM", "FPKM", "LOG")
`--batch`	Include path and filename of dataframe with batch normalization parameters
`--sjdbOverhang`	Sequencing platform read-length for constructing splice-aware reference previously (see 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: curateReference example

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

Table 2: Summary of dependency software, accession location, and purpose in the XPRESSpipe package.

Package	Purpose	Reference
fastp	Read pre-processing	(*11*)
STAR	Reference curation and read alignment	(*12*)
samtools	Alignment file manipulation	(*13*)
bedtools	Alignment file manipulation	(*14*)
deeptools	Alignment file manipulation	(*15*)
htseq	Read quantification	(*16*)
fastqc	Quality Control	(*17*)
multiqc	Quality Control	(*18*)
DESeq2	Perform differential expression analysis	(*19*)
dupRadar	Measure library complexity	(*20*)
pandas	Data manipulation	(*21*)
numpy	Data manipulation	(*22, 23*)
scipy	Data manipulation	(*24*)
matplotlib	Plotting	(*25*)
xpresstools	Normalization and matrix manipulation	This paper

XPRESSpipe is built upon several pre-established software packages, listed in Table 2. These dependencies are included in any Docker images for XPRESSpipe; however, if installing manually, these packages will need to be installed by the user.

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`

Other sub-modules that handle a point in the middle of sequence processing need to be of appropriate file type, explained more in the help menu or documentation.

2.1.3 Reference Curation

One of the first preparatory steps of RNAseq 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 the `modifyGTF` sub-module. Modifications include removing all genes not annotated as protein-coding and/or retaining only the longest transcript for each gene. For the purposes of ribosome profiling, where 5' and 3' transcript biases are frequent (*8,9*), the 5' and 3' ends of each gene record can be trimmed using the same function. While each of these steps are available as stand-alone sub-modules, all reference files used by XPRESSpipe, including STAR files and modified transcriptome files can be created by running the `curateReference` command. An example is shown below for creating a ribosome profiling-ready reference XPRESSpipe directory. The following assumes one is avoiding mapping to the first 45 nucleotides and last 15 nucleotides of the longest transcript for each gene, will only quantify to protein coding regions, and is tailored for mapping 50-bp single-end RNA-seq reads. 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.

Listing 2: curateReference example

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

2.1.4 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 pipelines. 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. **Trim**: First, reads need to be cleaned of artifacts for the library preparation and sequencing process. These include adaptors, unique molecular identifier (UMI) sequences, and base calls with low confidence. By doing so, non-native sequences are removed and reads can align properly to the reference sequence.

XPRESSpipe uses fastp, a faster, more accurate trimming package that has improved alignable read output (*11*). Adaptor sequence, base quality, and read length are all adjustable parameters available to the user.

2. **Align**: After trimming, reads are then aligned to a reference genome. XPRESSpipe uses STAR, which, while taking a memory intensive approach, is fast and one of the most best performing sequence alignment software packages currently available (*12,26*). XPRESSpipe performs a two-pass alignment of reads wherein first splice junctions are mapped and built into the reference; and second, reads are mapped accounting for these junctions to improve mappability over these regions. A sorted-by-coordinate and indexed BAM file is output and only unique alignments pass on to the next steps. PCR duplicates are detected and marked or removed for downstream processing. Optionally, bed and bigwig files can also be output.

3. **Count**: XPRESSpipe quantifies read alignments for each de-duplicated input file using htseq-count (*16*). 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). By default, htseq behavior conforms to TCGA standards by being strand agnostic, mapping assuming reads were sorted by name, and by using the `intersection-nonempty` method for handling reads overlapping multiple genes.

4. **Normalize**: Methods for count normalization are available within XPRESSpipe. For normalizations involving transcript length, the appropriate GTF (transcripts-only recommended) 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. 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) normalizatin, as outlined in Equations 1-4 (*27*).

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

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

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

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

5. **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 plots are output for all samples in a given experiment.

    - **Read Length Distribution**: Per sample, the lengths of all reads are analyzed by FastQC (*17*). 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.
    - **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. The analysis is performed using the PCR duplicate-tagged BAM files output by XPRESSpipe for each sample by dupRadar (*20*). Duplicate tagging is performed by samtools (*13*).

- **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 *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:

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

  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.

- **Codon Phasing/Periodicity Estimation Profile**: In ribosome profiling, a useful measure of a successful experiment comes by investigating the codon phasing of ribosome footprints (). 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 (). *p* is the distance from the start, *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. The same inputs are required as for the `metagene` sub-module, and the penalty is calculated in the same manner.

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

$$p^{-} = S \ - \ (L_{\mathrm{e}} \ + \ 16) \tag{8}$$

2.1.5 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.

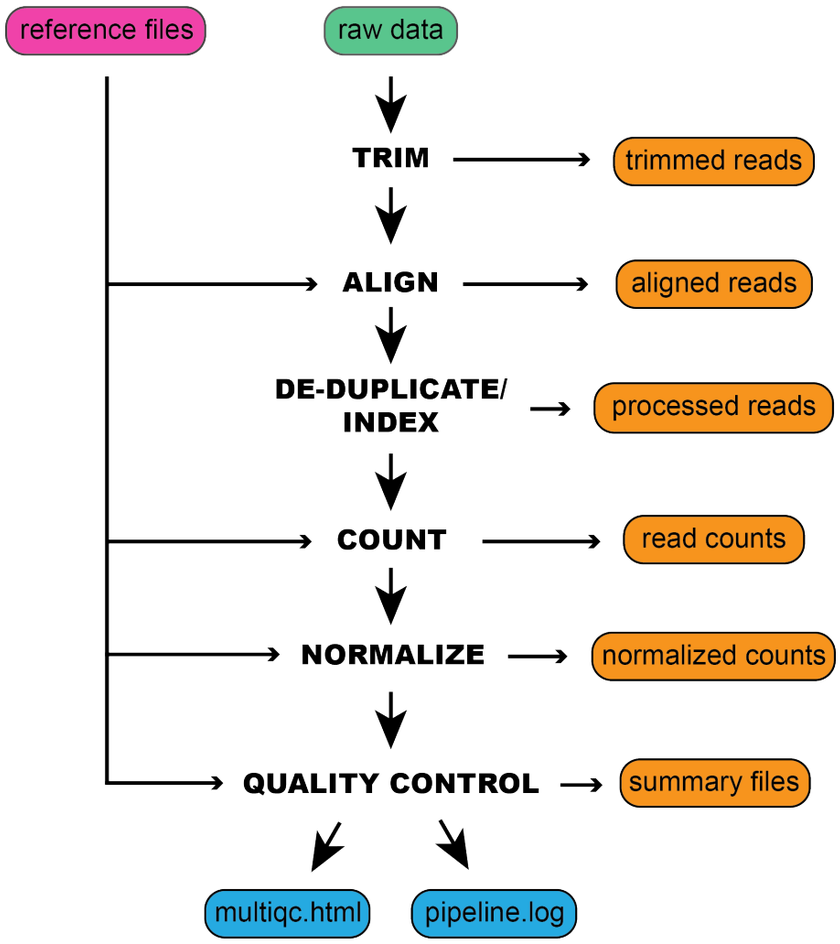

Figure 1: An example schematic of the inputs required by XPRESSpipe and organization of the ouputs.

2.1.6 Analyses

XPRESSpipe has two analytical sub-modules. The first acts as a wrapper for the R package, DESeq2 (*19*), to perform differential expression analysis on their data. More information on this sub-module can be found in the documentation (https://xpresspipe.readthedocs.io/en/latest/).

The second analytical tool introduced in XPRESSpipe is `rrnaProbe`. 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` works on a directory containing fastqc (*17*) zip compressed files to detect over-represented sequences for each sample. These sequences are then collated to create consensus fragments. A rank ordered list of over-represented fragments within the appropriate length range to target for depletion is then output. A BLAST (*28*) 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.

2.2 XPRESStools

XPRESStools is a python library of analysis features that builds upon existing packages, such as Matplotlib (*25*) and Seaborn (*29*) 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://xpresstools.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.

2.2.1 Getting Data

XPRESStools is designed for analyzing gene expression data, but it is tractable to most data types assuming two conditions are met:

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).

Several functions are built into XPRESStools for importing and formatting this data. Options include importing microarray and RNAseq expression data and metadata from the GEO database, as well as custom data and metadata that follows required formatting requirements. Several methods for normalization, such as RPM, RPKM, FPKM, TPM and log-scale normalization are accessible within this toolkit.

2.2.2 Analyzing Data

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

- **Principle Components Analysis**: Principle components analysis (PCA) for the data matrix are performed with Python's scikit-learn package (*30*) and desired principle components are plotted in a scatter plot via the matplotlib (*25*) and seaborn (*29*) packages. The XPRESStools PCA module, as in many other analysis modules within XPRESStools, 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 (*22, 23*) functionality 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

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

- **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, XPRESStools 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 (*24*). 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.

2.3 Availability

The source code for these packages is 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. XPRESStools is pip installable. XPRESSpipe is available as a Docker image. 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 (*31*).

2.4 Cost Analysis

Perform an example run using Docker image on AWS and give a cost per sample and time per sample output.

3 Results and Discussion

3.1 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.

3.1.1 Ribosome Profiling Dataset

In this section, we will look at the count values available on GEO and the counts we get out of the pipeline. Fig 2A will be correlations showing tight relationships between all mapped genes. F2B-D will replicate the relevant

figures from the paper.

3.1.2 TCGA Dataset

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.

3.1.3 Ease of Use Case Study

In this section, we will discuss a simple case study of 2-4 users with no computational experience and the time it took to get output and the quality of the output.

3.2 New Insights into Metabolism

Using the previously discussed ribosome profiling dataset, we will discuss new metabolic insights obtained by focusing on something other than what the authors were interested in. THe way this is explained will be fairly important – do we want to highlight that this is capable of finding something new (I don't know that that is really the purpose of this), that this just makes it so someone can quickly access the data and re-analyze it for their points of interest? Highlight again the amount of sequencing data out there and how most have only be used to explore one narrow aspect of biology. Figure 3 will include the insights gained. A potential Fig4 discussing validation of the insights.

4 Conclusions

We have described herein a new software suite, XPRESSyourself, a collection of tools and pipelines to aid in expression data processing and analysis. The software was designed to be flexible and modular so that in the future, new modules and tools can be easily added to and tested in the software suite. Discuss results of validation and biological study

Acknowledgments

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

Conceptualization	J.A.B.
Supervision	M.T.H., J.G., A.R.Q., J.P.R.
Project Administration	J.A.B.
Investigation	J.A.B.
Formal Analysis	J.A.B.
Software	J.A.B., J.R.B.
Methodology	J.A.B., J.R.B., M.T.H.
Validation	J.A.B., J.T.M., A.J.B., Y.O.
Resources	J.A.B., J.P.R.
Funding Acquisition	J.A.B., J.P.R.
Writing - Original Draft	J.A.B.
Writing - Review & Editing	J.A.B., M.T.H., J.G., A.R.Q., J.P.R.
Visualization	J.A.B.

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. M. Taschuk, G. Wilson, *PLoS Comput Biol* **13** (2017).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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