# 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 remove some of these barriers for users 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 where features can 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.**

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 as found itself the forerunner technology for highest quality 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 these technologies, 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tools used in processing RNA-seq data can be inhibitory if users are not aware of the proper tools to use or use outdated software (*3, 4*). For the experienced user, developing robust pipelines that process and check datasets can be laborious and introduce variability to data processing.

While several pipelines have emerged over the last several years that have been built to tackle various aspects of these bottlenecks, many suffer from usability issues, or are not easily modifiable. 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 these 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 five aspects in order to create an easy, reliable tool where large barriers-to-entry would be elimiated. These were create a tool that was useful, usable, reliable, efficient, and flexible (*10*).

With 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, and tools 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 hidden to maintain TCGA alignment standards. 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:

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

Listing 1: curateReference example

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*) |
| dupRadar | Measure library complexity | (*19*) |
| pandas | Data manipulation | (*20*) |
| numpy | Data manipulation | (*21,22*) |
| scipy | Data manipulation | (*23*) |
| matplotlib | Plotting | (*24*) |
| 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", ".txt"
- Paired-end reads should end in ".read1/2.suffix" or ".r1/2.suffix"
- 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 this files cannot contain any other files ending in ".txt" or ".fa"

#### 2.1.3 Reference Curation

One of the first preparatory steps of RNAseq alignment is curating a reference transcriptome for the aligner to map reads to. For the purposes of XPRESSpipe, a STAR reference should be created. However, for the purposes of several meta-analyses and more specific mapping, a STAR reference alone is not enough. For example, if one is aligning ribosome profiling reads and wishes to avoid mapping reads to the first 45 nucleotides in order to not quantify the inherent 5' biases in library preparation, a truncated transcriptome reference needs to be used when counting reads to transcripts. Additionally, if one wishes to only quantify reads to protein coding portions of the transcriptome, a custom reference file needs to be created. While each of these steps are available as stand-alone sub-modules, all reference files used by XPRESSpipe, including STAR files, modified transcriptome files, and other meta-analysis 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.

```
$ xpresspipe curateReference -o /path/to/output/location/
                             -f /path/to/fasta/genome/files/
                             -g /path/transcripts.gtf
                             --truncate_5prime 45
                             --truncate_3prime 15
                             --sjdbOverhang 49
                             --protein_coding
```

Listing 2: curateReference example

While current available arguments are limited, the design is simple enough where arguments could be added easily to give more control over STAR reference creation; however, current setting align with TCGA standards.

#### 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.

1. **Trim**: The first step in read processing is removing reads of artifacts for the library preparation process. These are most often adaptor sequences that were ligated to each read during the preparatory steps, which will not have homology to the organism's reference sequence. As sequencers are only so accurate, another common artifact are base calls that are statistically not confident. Therefore, in the first step of read processing, it is important to remove these sequences from each read to allow for proper alignment of reads to the reference. XPRESSpipe begins by trimming reads of adaptors, low quality bases, and reads that are too short using fastp, a more recent, faster trimming package that has improved alignable read output. Adaptor sequence, base quality, and read length are all adjustable parameters available to the user. Descriptions of the options can be found in Table 1.

2. **Align**: After sequencing reads have been trimmed of artifacts, the next step is to determine what transcripts these reads originated from to quantitate expression levels. This is done through alignment of each read to the reference transcriptome. XPRESSpipe uses STAR for this process as it has consistently proven itself as a fast and accurate alignment tool, such that it has been chosen as the TCGA standard alignment software. Alignment is made easy by generating a reference directory using XPRESSpipe's "curateReference" sub-module. XPRESSpipe then performs a two-pass alignment to maximize alignment to splice junctions. In the first pass, STAR identified potential splice junctions for each sample, generates a reference taking these sites into account, then uses this reference to remap reads to the reference. Afterwards, some file processing is performed to generate the appropriate output files and formats for downstream steps.

3. **Count**: Once reads have been aligned and genome coordinates have been determined, these reads need to be quantified. XPRESSpipe uses htseq for this purpose. In this step, reads coordinates along the genome are mapped to the reference transcriptome. If a coding-only or truncated reference was created during the reference curation, this will be used for the quantification of transcipts. 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. **Normalization**: As RNA-seq measurements are only relative, they are subject to sample and batch effects. These can arise from different people preparing libraries, one person preparing libraries on different days, or

are just inherent in differences in total reads per sample a given chip sequences. In order to remove sample effects arising per sequence chip, it is important to perform normalization. Reads-per-million (RPM) normalization controls for these sample effects. Reads-per-kilobase-million (RPKM) or Fragments-per-kilobase-million (FPKM) performs the same RPM normalization, but additionally normalizes for transcript length as longer transcripts will appear to have more reads aligning to these transcripts normally. Batch effect normalization controls for library to library variances. These normalization options are available in the XPRESSpipe arguments for each pipeline (see Table 1).

Normalization is performed based on the following equations:

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

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

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

#### 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 **??** provides an example of output file scheme for XPRESSpipe.

#### 2.1.6 Quality Control

It is important to perform quality control of sequencing sample to ensure they are reliable and their biological insight can be trusted. XPRESSpipe performs a variety of quality control measures. fastqc runs a gambit of quality control measures and multiqc summarizes these and metrics from trimming, aligning, and so on. The "readDistribution" sub-module will take the read size distributions for each library, as this can be informative, especially in the case of ribosome profiling, to ensure the proper read length was enriched in the sequencing library. The "metagene" sub-module will summarize read distribution along a representative transcript to highlight any sequencing biases, such as an enrichment of reads on the 3' end indicative of poor library preparation. The "periodicity" sub-module will take the most enriched read length from ribosome profiling footprint libraries and map the P-site of the ribosome to investigate the 3 nucleotide phasing characteristic of the translating ribosome. Each of these sub-modular quality functions will summarize their results in a single PDF for quick reference by the user. Details on how each of these methods were designed follows.

- **readDistribution**: Each ".fastq" sample file in the designated directory is processed by FastQC to generate general quality control statistics. The pipeline will take trimmed reads, which we find to be the most informative stage, but can be run on a ".fastq" or similar file at any stage of processing. Each FastQC output file is then searched for the read distribution statistics and a summary PDF with each sample's read distribution is generated using the Python libraries, Pandas and Matplotlib. This provides the user with a publication quality summary of each sample that can be rapidly analyzed by the user to assess general quality of each sample, especially in the case of ribosome profiling where generating footprint libraries within a size window is paramount.

- **metagene**: Each "SAM" sample file in the designated directory is processed by picard's CollectRnaSeqMetrics function. Each output file is then searched for the metagene statistics and a summary PDF with each sample's metagene profile is generated using the Python libraries, Pandas and Matplotlib so that the user can rapidly detect any biases or other issues in each sample library.

- **periodicity**: Each "SAM" sample file in the designated directory is processed to identify the dominant aligned read length per each sample. The SAM file is then modified to only contain the sequences of the dominant length, converted into an indexed BAM file using samtools, and analyzed by plastid's metagene function to determine P-site phasing of footprint samples in a ribosome profiling library. Each output file is then searched for the phasing statistics and a summary PDF with each sample's periodicity profile is generated using the Python libraries, Pandas and Matplotlib so that the user can rapidly determine phasing for each sample. This metric can be informative in ribosome profiling to ensure ribosome behavior is normal and that sufficient sequencing depth was obtained to create a robust ribosome profiling library.

#### 2.1.7 Analyses

XPRESSpipe has two analytical sub-modules. The first simplies the R package, DESeq2, for the user and acts as a command line interfacable option to access this statistical package.

The second, new 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 intervention, depletion of these sequences is often desired in order to have better depth of coverage of non-rRNA 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 to create ribosome footprints that commercial depletion kits won't detect and polyA selection kits are inoperable as footprints will not have the requisite polyA sequence. To this end, custom rRNA probes are recommended. "rrnaProbe" works on a directory containinng fastqc zip files to initially detect over-represented sequences. It then collate these sequences to determine consensus fragments by insert Jon's methods here. A rank ordered list of over-represented fragments within the appropriate length range to target for depletion is then output. BLAST capability to verify over-represented consensus fragments are indeed rRNA sequences is not incorporated, so any sequences that will be used as probes should be BLAST-verified.

### 2.2 XPRESStools

XPRESStools is a python library of analysis features that builds upon existing packages, such as Matplotlib and Seaborn to generate flexible, specific analyses frequently used by biological researchers that can each be executed in a single line of code. Additionally, many of the introduced analyses are currently available in the R environment but libraries in Python, a programming language becoming more and more common in biological research, have not yet been developed. A summary of these tools is summarized below and methods are discussed in subsequent sections. For the sake of brevity, methods for new capabilities in the Python toolkit will be discussed in detail. 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.

- **Reference File Modification**: GTF transcriptome reference files form the backbone of gene expression quanitification and analysis. However, as previously discussed, these files as provided are often not customized or easily modified. To address this issue, we developed a Python tool for modifying these files to create GTF files with coding only records, or records where each exon 1 is truncated by n nucleotides to avoid mapping to regions where library biases may exist. Additionally, these reference files are used throughout the XPRESSyourself framework for gene normalization and data matrix reformatting.

- **Microarray Probe Collapse**: When performing a microarray experiment, there are often multiple probes for a given gene and probes that will map to several transcripts. It is often desireable to obtain a matrix of microarray data with common gene names where these probes are condensed. While several R packages exist to solve this issue, no Python packages are available. As microarrays still have incredible utilityreference of RNAseq and microarray correlation, we thought it relevant to create a Python based package to handle this formatting of microarray data matrices.

- **Principle Componenet Analysis**: Principle component analysis (PCA) is a frequent tool in expression analysis to determine if samples of different backgrounds will cluster based on their expression to show that they are truly biologically distinct. Standard R packages for performing PCA will optionally plot confidence intervals to convey statistical significance of this delineation of sample types; however, nothing comparable is available as a streamlined Python package for biological research. We thus developed a package where these confidence intervals are plotted over the PCA scatterplot.

- **Volcano Plots**: Volcano plots are a frequently used analysis in gene expression, proteomics, and metabolomics experiments, to name a few. These plots are a simple way to convey both magnitude and significance of change in expression or other metric of experiment versus control conditions. We developed a Python package that will plot this data, and will additionally plot thresholds, highlight genes of interest by color-coding and name label, and will output significant hits for further analysis.

#### 2.2.1 Getting Data

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

1. **Expression Matrix**: It is assumed the data matrix is *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 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 and treatment).

Several functions are built into XPRESStools for importing and formatting this data. Options include imported microarray and RNAseq expression data and metadata from the GEO database, as well as custom data and metadata that follows required formatting.

### 2.2.2 Normalizing and Formatting Data

Normalization and formatting options are available for both RNAseq and microarray processing. RNA-seq sample and batch normalization procedures and methods are discussed above. Gene normalization is performed using a standard scaling procedure via scikit-learn, where each gene is scaled so that the mean is 0 and standard deviation is 1 to faciliate gene-to-gene comparisons.

Microarray probe collapse is designed to allow for conversion of probe IDs to common names and collapse of multiple probes. A user will download the relevant GPL or similar text file that includes information for each probe for a given microarray platform and the genes to which the probe maps. There are three possibilities when it comes to probe mapping that the probe collapse procedure will handle:

1. **Probe maps uniquely**: In such a case, the probe ID is converted to the common gene name found in the probe index and no further manipulation is performed.

2. **Probe maps to multiple targets**: In these cases, the user has two choices. They can either drop the probe from the analysis or leave as is and convert the probe ID to reflect all genes it maps to.

3. **Several probes map to a single gene**: In these cases, the probe collapser will group these probes together and for each sample take the average measure of these probes. Probe ID will be converted to common gene name.

To show consistency and accuracy of this Python based probe collapser, we benchmarked it against the commonly used software package, Affy. Insert figure that compares this to Affy

### 2.2.3 Analyzing Data

While a litany of analysis tools are included in XPRESStools as of 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 Component Analysis**: Principle components for the data matrix are performed with the Python sklearn package and desired principle components are plotted in a scatter plot via the Seaborn package. 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 by the following steps:

    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

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

- **Volcano Plotting**:

When these computations are performed with a metadata table containing several sample types, these computations will be performed for each label type individually and plotted on the scatter plot.

#### 2.2.4 Other Features

How the GTF truncation works

### 2.3 Unit Testing and Code Coverage

As this suite of tools is intended to be flexible, and built upon in the future, unit tests to cover general and edge cases with each function have been created and future developments to the pacakge will require new functional tests to maintain package integrity.

### 2.4 Availability

The source code for these packages is open source protected under the GPL-3.0 license and can be publically accessed at https://github.com/XPRESSyourself. Updates to the software are version controlled and maintained on GitHub. Packages are pip and/or conda installable in addition to source installable.

## Results and Discussion

### 2.5 Example Data Walkthrough

In order to describe the utility of the tools available in this software suite, we prepared a Jupyter Notebook with an example walkthrough for the data analysis of an example dataset. This notebook can serve as a guide and framework for future analyses of data by a user of the software. We chose to use the publicly available microarray dataset, GSE20916, which investigates the gene expression in the colon and associated tumors. While this analysis is for a microarray dataset, code blocks describing and executing pertinent steps for RNAseq are included in this notebook. In this dataset, the normal colon and adenoma tissue are selected and imported, cleaned, and normalized. In this example, the microarray probe collapser is used to remove multi-mapping probes and average duplicate probes for a given gene. After data has been formatted, normalized, and checked for overall quality, a simple, example analysis of the biology behind the data is presented. In this example, we chose to investigate mitochondria and metabolism associated factors and their potential role in the development of normal colon tissue into adenomous tissue that could lead to tumorous tissue. In this analysis, we showed that the transporter, *MPC1* ... CHANGE THIS TO RIBOSOME PROFILING DATA AND FIND SOMETHING NEW

### 2.6 Cost Analysis

### 2.7 Benchmarking

TCGA

Ribosome profiling

Affy vs collapser

## 3 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.

Easy

New

Reproducible, standardized – ref TCGA benchmark

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

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

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

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

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

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