ribofootPrinter: A precision python toolbox for analysis of ribosome profiling data

Nicholas R. Guydosh

Laboratory of Biochemistry and Genetics, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD, 20892

Contact: nicholas.guydosh@nih.gov

Keywords

Ribosome profiling, reading frame, metagene, pausing, uORF, iORF, dORF, python, MANE

Abstract

Ribosome profiling is a valuable methodology for measuring changes in a cell's translational program. The approach can report how efficiently mRNA coding sequences are translated and pinpoint positions along mRNAs where ribosomes slow down or arrest. It can also reveal when translation takes place outside coding regions, often with important regulatory consequences. While many useful software tools have emerged to facilitate analysis of these data, packages can become complex and challenging to adapt to specialized needs. In particular, the results of meta analysis of average footprint data around sequence motifs of interest can vary substantially depending on the normalization method that is utilized. We therefore introduce ribofootPrinter, a suite of Python tools designed to offer an accessible and modifiable set of code for analysis of ribosome profiling data. Footprint alignments are made to a simplified transcriptome, keeping the code intuitive, and multiple normalization options help facilitate interpretation of meta analysis, particularly outside coding regions. We believe this tool has promise to carry out sophisticated analysis yet offer simplicity to make it readily understandable and adaptable.

Introduction

Ribosome profiling (also referred to as Ribo-seq) has become an established approach for revealing which genes are translated and how ribosomes are distributed along transcripts in cells [1, 2]. Ribosome profiling data are generated by deep sequencing cDNA libraries that are generated from ribosome-protected mRNA "footprints" that typically measure 27-29 nt in length. However, in some experimental conditions, footprints can measure ~16 nt, ~21 nt, or >29 nt [3-6] and reveal additional information. To analyze ribosome profiling datasets, reads are typically aligned to the transcriptome and downstream software packages can be used to reveal a rich and precise view of a cell's translational program. Analysis can show the distribution and loading (translational efficiency) of ribosomes across messenger RNAs (mRNAs). It can also offer insight on where actively-translated open reading frames (ORFs) exist, including within non-canonical regions, such as 5' and 3' untranslated regions (UTRs) and out-of-frame sequences within coding sequences (CDSs) [7]. Related approaches [8-10] for footprinting disomes (disome-seq) or 40S ribosomal subunits (TCP-seq) [11, 12] have enhanced the approach to address questions related to ribosome collisions and translation initiation.

Many capable software tools have emerged to analyze ribosome profiling data, in part because no single approach tends to fit every need. However, one of the challenges faced by all approaches is how to handle the complex nature of mRNA splicing patterns in the cell. In humans, most transcripts are spliced into multiple isoforms and this creates the problem of how to represent a particular gene's level of ribosome occupancy, particularly when different isoforms (gene models) are translated with varied efficiency. Several solutions have emerged, including use of masking functions to exclude transcript regions where splicing patterns are variable [8]. Other approaches simplify the problem by mapping to the longest isoform or

isoform with the most exons [13-15]. Another challenge posed by splicing is the computational burden of converting between genomic and transcriptomic coordinate systems [10]. This complexity also tends to make the code difficult to understand, raising barriers to code modification for beginners.

One of the most common forms of analysis is to create a "meta" plot by averaging together ribosome profiling data around particular sequence motifs of interest (i.e. metagene or metacodon analysis). The simplest form of this analysis involves computing the arithmetic mean (or median) for read counts at every nucleotide position surrounding the feature of interest. However, the downside of this approach is that every feature (i.e. gene or codon) is not weighted equally in the average; highly expressed features tend to dominate. The need to consider alternatives to normalization is becoming more critical as the importance of translation outside conventional coding sequences has become clear [7, 16, 17]. In such cases, the most appropriate method of normalization may not be immediately obvious and it is therefore advantageous to consider multiple forms of normalization.

A final challenge of many computational tools is the loss of support over time and their eventual deprecation as changes are inevitably made to supporting packages and the programming languages themselves. This problem is particularly acute for packages that are complex and rely on extensive outside software to function. Therefore, simple and readily modifiable tools are advantageous for promoting longevity and continuity.

We therefore introduce ribofootPrinter, a precision Python toolbox for analysis of ribosome profiling data from any organism. The package simplifies the splicing problem by relying on a simplified transcriptome and offers several functions that can be readily modified. Importantly, the software also offers the option of multiple normalization approaches for meta

analysis and can be further expanded. The simplicity of the approach makes this package appealing for both introductory programmers as well as users interested in a straightforward platform. ribofootPrinter is particularly useful for understanding several topics of growing importance, such as finding small ORFs, assessing ribosome stalling at sequences of interest, and evaluating translation levels outside canonical ORFs. The tool should work well with not only 80S conventional ribosome footprinting, but also 40S and disome footprinting approaches.

Implementation

The first step (Fig. 1A) is to map ribosome footprint data to a simplified transcriptome with bowtie [18]. Recently, the Matched Annotation from the NCBI and EMBL-EBI (MANE) project offers a curated transcriptome for *H. sapiens* with a single recognized (most likely to be used) isoform per gene. Alternatively, top transcript isoforms from RefSeq Select, can be utilized. While this simplification step represents an obvious compromise, it greatly facilitates downstream analysis, by allowing faster performance and eliminates artifacts associated with using the longest (but not most common) transcript isoform or when eliminating common genes due to ambiguous splicing. This curation process also avoids artifactual peaks common in pseudogenes and non-coding RNAs. Moreover, the elimination of code to handle splicing and conversion to genome coordinates makes it easier to understand for programmers of all levels. All the code is available online and described in more detail below. It is designed to run with Python 3 and requires the BioPython package [19] for sequence manipulations. All of the package's functionalities typically run within a few minutes on modern personal computers.

Prior to creation of bowtie indexes, the transcriptome annotation should be simplified to match that as described in Methods. Critical information, including gene name, alias, sequence,

and annotated positions of UTR regions are included. Following bowtie alignment, the first step in the ribofootPrinter pipeline is to create data structures to hold the ribosome occupancy data. These structures are created using python dictionaries (Fig. 1B), with keys for each gene, to include both information, such as gene name, sequence, and UTR positions, and footprint data. In this way, these occupancy files keep metadata and experimental data together. As with other tools [8], information is saved for both 5' and 3' end assignment of mapped reads. As shown previously, 3' end assignment often offers superior reading frame alignment for ~28-nt reads from ribosomes in yeast cells, while 5' end alignment is superior for 40S subunits in yeast or 80S ribosomes in many human cell types [4, 17, 20]. The data structure can also be modified to organize read data by footprint length.

With these ribosome density files in hand, several tools are available for analysis (Fig. 1A). One of the most basic applications is to output read occupancies as a function of nucleotide position for a particular gene model. This is accomplished with the writegene2 package. The output data in reads per million mapped reads (rpm) is provided in csv format can then be used to create images of ribosome occupancy on a spliced transcript. Like transcript-centric viewers [21], this script offers a distinct advantage over traditional genome viewers that include intronic regions (Fig. 2A, example of *GAPDH* gene model).

Metagene and total quantitation

The package offers the ability to perform average ("metagene") analysis at start or stop codons of coding sequences (Fig. 2B). This analysis at start codons is essential to estimate the position of the ribosome P site with respect to either the 5' or 3' end of mapped footprints. This "shift" value is essential for all other analysis in the package. Typically, 5' assigned ribosome

profiling in either yeast or human cells results in footprint peaks 12 nt upstream of start codons, showing that the first base of the P site is about 12 nt away from the 5' end. Metagene analysis also offers a visual metric of reading frame fidelity by revealing how strong the periodicity is in the coding region. It can also visually reveal the level of translation in UTR regions, and directly show how it compares with the CDS. Traditional metagene analysis is performed using an arithmetic mean to compute the average. However, this overweights the most highly expressed genes in the transcriptome and can result in artifacts. Therefore, an alternative with equal weighting of all genes, is made available. Weighting is performed by counting total reads within the coding sequence of the gene, subject to thresholding options, and normalizing the data to this prior to averaging.

The package also includes the ability to quantitate total read counts per gene, reported as both raw reads for differential expression analysis tools, such as DESeq2 [22], or normalized to total mapped reads and length (in kilobases) of gene models (rpkm). Outputs are offered for both coding sequence and UTR regions to give a quick portrait of the level of 5' and 3' UTR translation levels, values that are known to change due to alterations in the fidelity of initiation or recycling, respectively. Here, on sample data from WT cells, footprint levels in 5'UTRs are considerably greater than in 3'UTRs as is generally expected (Fig. 2C-D) [7]. An optional mode outputs reads in each of the 3 reading frames of the coding sequence, allowing visualization of possible frame defects that can result from events such as frameshifting or leaky scanning (Fig. 2C).

Small ORF analysis

Accurate quantitation of ribosome footprints on small open reading frames outside coding sequences represents an active area of investigation. These upstream open reading frames (uORFs) in 5'UTRs and downstream open reading frames (dORFs) in 3'UTRs are emerging as important regulatory elements that can either enhance or inhibit translation of the main coding sequence [23, 24]. While metagene and total quantitation across a UTR region can offer some insight about translation in UTRs, precision estimation of a particular small ORF's usage is more complicated, particularly because small ORFs tend to be overlapping and may initiate with multiple start codons. In addition, initiation may occur with a near-cognate start codon (1-bp mismatch from ATG). In ribofootPrinter, quantitation of reads on small ORFs is implemented by counting in-frame reads only on all potential small ORFs, thereby excluding most reads from overlapping out-of-frame ORFs. The small ORFs that are analyzed can be filtered by start codon (perfect match or 1 mismatch) and by length, since longer ORFs will be less subject to noise. The output of this analysis can then be combined with appropriate thresholding analysis to identify likely expressed small ORFs. On a sample dataset, ribosome density across uORFs reveals many uORFs with expression levels comparable to that of the main ORF (Fig. 2E).

Metasequence analysis and pause scores

Average analysis of sequence-specific positions within gene models ("metasequence" or "metacodon" analysis) is critical for estimating levels of ribosome stalling during elongation or usage of small ORFs in UTRs (uORFs and dORFs) or internal to coding sequences (iORFs). The possavg package therefore can accept a specific nucleotide or amino-acid sequence and then compute average ribosome footprints within a defined window around these motifs in the transcriptome. Averages are performed subject to thresholding over the window of interest. This

analysis can be used to check particular motifs for tendency to slow elongation in coding sequences and shows that when footprints are averaged around in-frame Pro vs Ser in the P site, Pro clearly slows the ribosome (Fig. 3A), as is expected [25]. Importantly, when this analysis is carried out for UTR regions, it can offer a meta analysis of small ORF translation. Critically, the average can be locally normalized to the window around the motif or normalized to the gene model's respective coding sequence. For analysis of uORFs, the outcome is similar with both methods since ribosome occupancy on uORFs is often comparable to that of coding squences (Fig. 3B, left). In contrast, ribosome occupancy on dORFs is quite low compared with coding sequences and therefore the locally normalized analysis clearly reveals translation on dORFs while the CDS-normalized analysis reflects the very small level of translation (Fig. 3B, right).

Last, the code can also compute a pause score from the meta analysis plots. The score is computed by dividing the counts in the peak by the background level (Fig. 3C). Analysis for every single amino acid in the ribosome P site clearly shows how Pro is the slowest amino acid, as expected.

Analysis of pause score distribution

Finally, to gain insight about the distribution of pause scores, pause scores at individual mRNA positions can be computed, subject to thresholding of background levels. This analysis enables more advanced statistical analysis of the distribution of pause scores at motifs of interest and can show global trends. For example, it reveals that translation of Pro-Pro-Gly is slower than Ala-Ala-Ala globally (Fig. 3D).

Conclusions

We have introduced ribofootPrinter, a readily configurable software toolbox that is distilled down to essential functionalities for the analysis of ribosome profiling data. In particular, its approach of reducing mapping complexity by using a reduced transcriptome and offering meaningful tools for analysis of translation offers a powerful and easy-to-use resource for researchers in the field. In addition, its multiple options for meta analysis normalization enhance this capability. While no package serves the need of every user, we believe ribofootPrinter offers a useful balance of simplicity and key functionalities, particularly with regard to meta analysis, that will make it a good choice for many users. In addition, it offers a simple way to get ribosome profiling footprint occupancy and respective mRNA transcript sequences into standard Python list and string data types to help facilitate the development of new forms of analysis.

Several enhancements could eventually improve the capabilities of the software. To add more normalization options for meta analysis of positions of interest (posavg function), a future package could include averaging that computes the arithmetic mean. While this would not be equally weighted between sites, it has the advantage of being in units of rpm and therefore offers an intuitive result that is comparable to the type of analysis performed by other packages. It would also be desirable to include capabilities to find pause peaks and report the underlying sequences as potential pause sites. In addition, more sophisticated data structures could allow for the storage of variable read lengths to facilitate more precise analysis of read-length specific mapping of ribosome footprints. However, including data for every size quickly becomes overwhelming for data storage, though alternative binary approaches could resolve this [26].

We anticipate that the growing realization that translation outside of coding regions is critical for regulating gene expression and generates functional peptides will necessitate tools for

characterization of translation in UTRs and out-of-frame regions [16]. We therefore believe the utility of this software package will continue to grow and be adapted to incorporate new methods for studying these phenomena.

Methods

ribofootPrinter is available on Github:

https://github.com/guydoshlab/ribofootPrinter

The software is best run inside a Python 3 virtual environment (venv) where Biopython has been installed. We assume the user has appropriately pre-processed ribosome footprint reads, including steps to trim linkers, remove duplicates using unique molecular identifiers (UMIs), and subtract rRNA and other non-coding RNA sequences. At this stage, the reads can be aligned to the reduced transcriptome of choice using bowtie. For optimal study of reading frame, we recommended use of 0 mismatches during alignment. However, 2 mismatches can be used to increase the depth of mapped reads. The fasta file used for generation of bowtie indexes must adhere to the following header format with all fields separated by "|" characters, as shown in this example:

```
>NM_002046.7|NM_002046.7|NM_002046.7|NM_002046.7|NM_002046.7|GAPDH|1285|UTR5:1-76|CDS:77-1084|UTR3:1085-1285
```

The first position should be the formal gene name, i.e. "`NM_002046.7`" followed by the alias gene name in the sixth position. This is followed by overall gene length. The final 3 positions should include the prefixes "`UTR5:`", "`CDS:`", and "`UTR3:`" followed by the start and end position of that region, separated by a hyphen. Once SAM files are obtained from the bowtie alignment, they can be converted into the stored read data structures using the `builddense_m` function in `mammalian_builddense.py`

Once the pickled and compressed data structures are generated (extension `pkl.gzip`), the software currently allows 6 forms of analysis to performed. All analysis routines are performed by modifying text input files with relevant parameters. These parameters are read by the code and a metadata output including error messages and the read-in parameters is generated alongside the data output. All data outputs are in csv format. Guidance for choice of input parameters is provided in the commentary and associated documentation on Github. Overall, there are 6 major analysis tools:

`writegene2` outputs ribosome profiling data aligned to a particular gene model.

`metagene` averages ribosome profiling data around start or stop codons.

`genelist` counts reads that map to CDS and UTR regions of gene models, and individual frames of the CDS.

`smorflist` counts reads that map in frame to uORF or dORFs in the 5' or 3' UTR, respectively.

`posavg` averages ribosome profiling data around any sequence feature of interest or computes a pause score for every occurrence of all 61 codons or 20 amino acids.

`posstats` computes pause scores for every individual occurrence of a sequence feature of interest.

Datasets and transcriptome

Sample data shown here were derived from WT A549 cells and previously published as AK006F and AK014F [17]. They are publicly available on NCBI GEO: GSE158044. Reads were aligned to a transcriptome created by downloading RefSeq Select+MANE

(ncbiRefSeqSelect) from UCSC on 14 April 2020 and removing duplicates on alt chromosomes (includes annotations from MANE ver. 0.6).

Funding

This research was supported by the Intramural Research Program of the NIH, The National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), DK075132 to N.R.G.

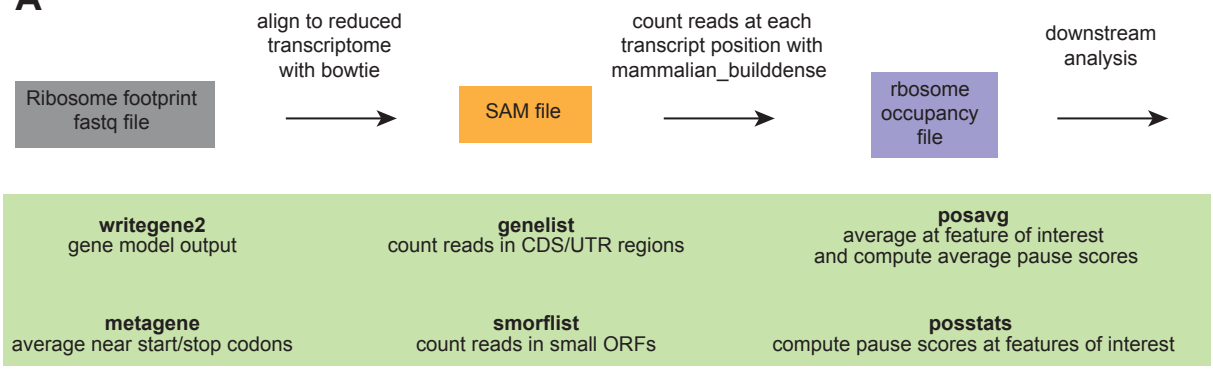
Acknowledgements

I would like to thank Agnes Karasik for help identifying bugs in early versions of the code and I thank the Guydosh Lab and James Marks for helpful discussions about the vision, development, presentation, and naming of the code.

Fig. 1

ribofootPrinter pipeline

A



B

Ribosome occupancy data structure

Fig. 1. A) Flow chart of steps to analyze ribosome profiling data with ribofootPrinter. Reads are first aligned to the reduced transcriptome with bowtie and the resultant SAM file is processed into a ribosome occupancy file. Data can then be analyzed with six downstream analysis tools. **B)** Data structure for storing ribosome occupancy data (both 5' and 3' assigned) and sequence metadata.

Fig. 2

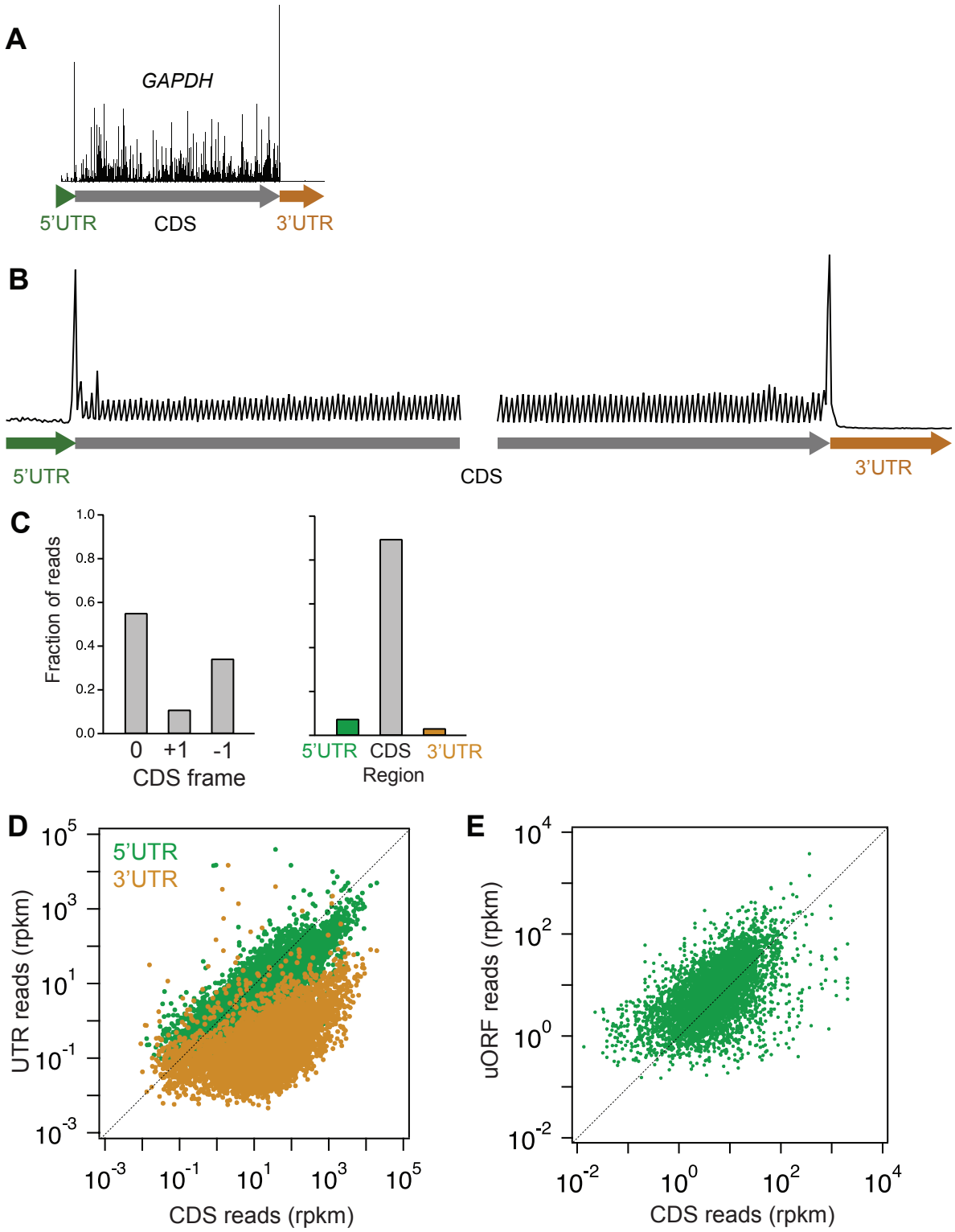

A
GAPDH
5'UTR
CDS
3'UTR
B
5'UTR
CDS
3'UTR
C
Fraction of reads
1.0
0.8
0.6
0.4
0.2
0.0
0
+1
-1
CDS frame
5'UTR
CDS
3'UTR
Region
D
5'UTR
3'UTR
UTR reads (rpkm)
CDS reads (rpkm)
E
uORF reads (rpkm)
CDS reads (rpkm)

Fig. 2. A) Example output of mapped ribosome profiling reads on a gene model for *GAPDH*. **B)** Metagene plot for start and stop codons reveals 3-nt periodicity in the coding sequence. It also shows low level translation in the 5'UTR and nearly no translation in the 3'UTR. **C)** Output of gene quantitation reveals (left) most reads map to the main, or 0, reading frame and smaller populations map to the +1 and -1 frames. Analysis (right) of mRNA regions reveals the vast majority of reads map to the CDS with a low level mapping to the 5'UTR and even less to the 3'UTR, consistent with (B). **D)** Data from (C, right) plotted at the level of gene models shows that 5'UTRs typically have more reads than 3'UTRs (each dot is 1 gene model). **E)** Result of small ORF analysis (each dot is 1 uORF) in 5'UTRs reveals that, as a function of CDS reads, density of reads on individual uORFs is comparable to the CDS (dots more or less along diagonal). This analysis of actual uORFs contrasts with the result in (D) where analysis of entire 5'UTRs reduces the apparent density due to inclusion of regions without uORFs (data further above diagonal in E vs D).

Fig. 3

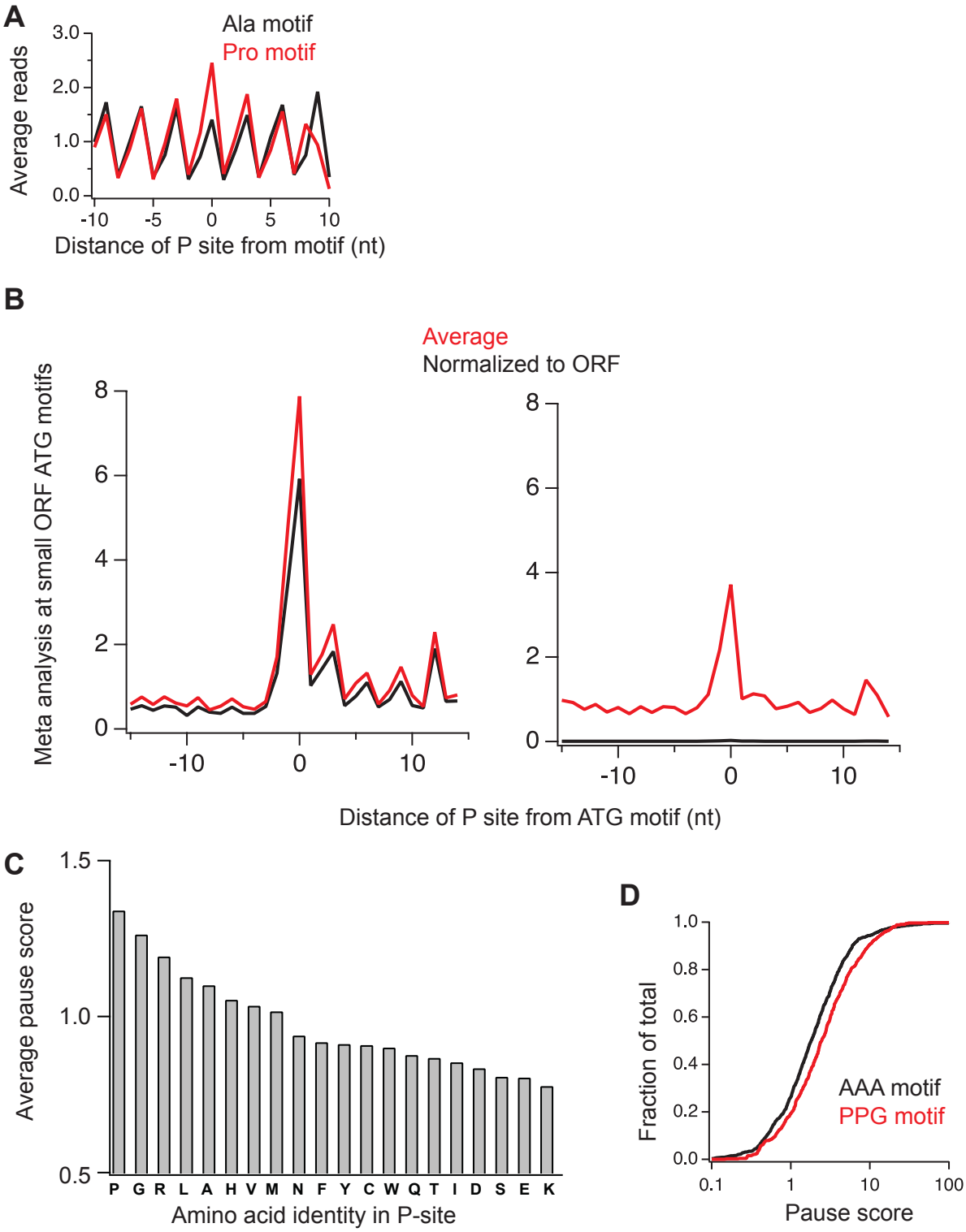

A
Ala motif
Pro motif
Average reads
3.0
2.0
1.0
0.0
-10
-5
0
5
10
Distance of P site from motif (nt)
B
Average
Normalized to ORF
Meta analysis at small ORF ATG motifs
8
6
4
2
0
-10
0
10
Distance of P site from ATG motif (nt)
C
Average pause score
1.5
1.0
0.5
P G R L A H V M N F Y C W Q T I D S E K
Amino acid identity in P-site
D
Fraction of total
1.0
0.8
0.6
0.4
0.2
0.0
AAA motif
PPG motif
0.1
1
10
100
Pause score

Fig. 3. A) Average analysis of reads at Pro and Ala codon motifs reveals a clear peak when the Pro motif (but not the Ala motif) is placed in the P site. **B)** Metaposition analysis of ATG codons in the 5'UTR (left) reveals a clear peak and increased ribosome occupancy and 3-nt periodicity after the peak. This result is consistent with efficient translation of uORFs. Whether data is normalized to a local window (red) or to the respective coding sequence (black) does not dramatically affect the result since uORF translation levels are comparable to CDS translation levels. Analysis of ATG codons in 3'UTRs (right) shows a similar peak when reads are combined with a locally-normalized average, showing dORF translation does take place. However, the lack of a peak when reads are normalized to the ORF (black) shows the translation level is quite low. **C)** Pause scores from positional average analysis at every amino acid reveals Pro induces the most stalling, a level nearly double above the least stall-inducing single amino acids. **D)** Comparable analysis of the distribution of all pause scores at AAA and PPG tri-amino acid motifs shows a clear difference in the overall distributions (tri-amino acid motif is placed in E, P, A sites).

References

1. Ingolia, N.T., et al., *Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling.* Science, 2009. **324**(5924): p. 218-23.
2. McGlincy, N.J. and N.T. Ingolia, *Transcriptome-wide measurement of translation by ribosome profiling.* Methods, 2017. **126**: p. 112-129.
3. Wu, C.C., et al., *High-Resolution Ribosome Profiling Defines Discrete Ribosome Elongation States and Translational Regulation during Cellular Stress.* Mol Cell, 2019. **73**(5): p. 959-970 e5.
4. Guydosh, N.R. and R. Green, *Translation of poly(A) tails leads to precise mRNA cleavage.* RNA, 2017. **23**(5): p. 749-761.
5. Lareau, L.F., et al., *Distinct stages of the translation elongation cycle revealed by sequencing ribosome-protected mRNA fragments.* Elife, 2014. **3**: p. e01257.
6. Schmidt, C., et al., *The cryo-EM structure of a ribosome-Ski2-Ski3-Ski8 helicase complex.* Science, 2016. **354**(6318): p. 1431-1433.
7. Mudge, J.M., et al., *A community-driven roadmap to advance research on translated open reading frames detected by Ribo-seq.* bioRxiv, 2021: p. 2021.06.10.447896.
8. Dunn, J.G. and J.S. Weissman, *Plastid: nucleotide-resolution analysis of next-generation sequencing and genomics data.* BMC Genomics, 2016. **17**(1): p. 958.
9. Liu, Q., et al., *RiboToolkit: an integrated platform for analysis and annotation of ribosome profiling data to decode mRNA translation at codon resolution.* Nucleic Acids Res, 2020. **48**(W1): p. W218-W229.
10. Tjeldnes, H., et al., *ORFik: a comprehensive R toolkit for the analysis of translation.* BMC Bioinformatics, 2021. **22**(1): p. 336.
11. Archer, S.K., et al., *Dynamics of ribosome scanning and recycling revealed by translation complex profiling.* Nature, 2016. **535**(7613): p. 570-4.
12. Meydan, S. and N.R. Guydosh, *Disome and Trisome Profiling Reveal Genome-wide Targets of Ribosome Quality Control.* Mol Cell, 2020. **79**(4): p. 588-602 e6.
13. Ishimura, R., et al., *RNA function. Ribosome stalling induced by mutation of a CNS-specific tRNA causes neurodegeneration.* Science, 2014. **345**(6195): p. 455-9.
14. Sendoel, A., et al., *Translation from unconventional 5′ start sites drives tumour initiation.* Nature, 2017. **541**(7638): p. 494-499.
15. Ahmed, N., et al., *Identifying A- and P-site locations on ribosome-protected mRNA fragments using Integer Programming.* Sci Rep, 2019. **9**(1): p. 6256.
16. Chen, J., et al., *Pervasive functional translation of noncanonical human open reading frames.* Science, 2020. **367**(6482): p. 1140-1146.
17. Karasik, A., et al., *Activation of the antiviral factor RNase L triggers translation of non-coding mRNA sequences.* Nucleic Acids Res, 2021. **49**(11): p. 6007-6026.
18. Langmead, B., et al., *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.* Genome Biol, 2009. **10**(3): p. R25.
19. Cock, P.J., et al., *Biopython: freely available Python tools for computational molecular biology and bioinformatics.* Bioinformatics, 2009. **25**(11): p. 1422-3.
20. Young, D.J., S. Meydan, and N.R. Guydosh, *40S ribosome profiling reveals distinct roles for Tma20/Tma22 (MCT-1/DENR) and Tma64 (eIF2D) in 40S subunit recycling.* Nat Commun, 2021. **12**(1): p. 2976.

21. Kiniry, S.J., et al., *Trips-Viz: a transcriptome browser for exploring Ribo-Seq data.* Nucleic Acids Res, 2019. **47**(D1): p. D847-D852.
22. Love, M.I., W. Huber, and S. Anders, *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.* Genome Biol, 2014. **15**(12): p. 550.
23. Hinnebusch, A.G., I.P. Ivanov, and N. Sonenberg, *Translational control by 5′-untranslated regions of eukaryotic mRNAs.* Science, 2016. **352**(6292): p. 1413-6.
24. Lin, Y., et al., *Impacts of uORF codon identity and position on translation regulation.* Nucleic Acids Res, 2019. **47**(17): p. 9358-9367.
25. Schuller, A.P., et al., *eIF5A Functions Globally in Translation Elongation and Termination.* Mol Cell, 2017. **66**(2): p. 194-205 e5.
26. Ozadam, H., M. Geng, and C. Cenik, *RiboFlow, RiboR and RiboPy: an ecosystem for analyzing ribosome profiling data at read length resolution.* Bioinformatics, 2020. **36**(9): p. 2929-2931.