Application of Antisense Technology in the Control of Upstream Open Reading Frame-Mediated Translation



Nenad Svrzikapa

DEPARTMENT OF PAEDIATRICS

LINACRE COLLEGE, UNIVERSITY OF OXFORD

Thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2022

Declaration

The research presented in this thesis was conducted in the laboratory of Professor Matthew Wood at the University of Oxford, Department of Paediatrics between the years of 2019 and 2022. This work was supervised by Professor Matthew Wood and Dr. Thomas Roberts. I, Nenad Svrzikapa, hereby declare that the work presented in this thesis is my own, and that I have credited my sources, and my colleagues with whom I collaborated on specific sections of this work. Namely, the destination psiCheck dual-luciferase reporter vector was designed by Dr. Thomas Roberts and Dr. Britt Hanson before I started work on this project. The initial gene panel was designed by Matthew Drake and Thomas Roberts. The poly-E gene panel presented together with the machine learning predictions were designed and screened by Dr. Britt Hanson. The simulated sequences predicted in the machine learning section, were designed by me, but were experimentally validated by Dr. Rushdie Abuhamdah. The silencing by uORF introduction programming code was developed by Vanessa McLoughlin, under joint guidance by Dr. Thomas Roberts and myself. This thesis includes figures and text from manuscripts which are currently under preparation. The figures and text presented in this thesis were generated in the timeframe specified above and have not been submitted for the attainment of a degree at this, or any other academic institution. The antisense oligonucleotides used in this study were designed by me. Where explicitly stated, the antisense oligonucleotides use published, and patented platform technology owned by Wave Life Sciences, Cambridge, MA, United States, and Ionis Pharmaceuticals, Carlsbad, CA, USA. My thesis was conducted under a Russell Studentship Agreement, with the generous support of Wave Life Sciences.

Acknowledgements

The completion of this thesis would not have been possible without the dedicated support of many remarkable people. I am truly humbled by this honor and the generosity extended to me.

Above all, I would like to express my sincere gratitude to my supervisors Prof. Matthew Wood and Dr. Thomas Roberts. For their kindness, thoughtfulness, creativity, scientific rigor, and inspiring pursuit of knowledge, I am profoundly grateful. Being a member of the Wood laboratory and being mentored by them has been a true privilege.

To my examiners Professors Stephen Hyde and Edvard Smith, who recognized me as someone worthy of this title, I owe tremendous thanks, for they set aside many hours to help me improve this body of work and gave me a pleasant and memorable viva experience. I would also like to thank my assessors Prof. Leonard Seymour and Prof. Jenny Taylor for their advice and guidance during the early development of this work.

The opportunity to embark on this journey would not have been possible without the generous support of Dr. Christopher Francis and Dr. Paul Bolno, two great leaders who are transforming the field of genetic medicines, and Prof. Gregory Verdine, who honored me with a recommendation and whom I hold in the highest esteem.

From the moment of its inception, this project has benefited from countless discussions, feedback, honest criticism, and encouragement from many close friends and colleagues, most notably: John and Faye Waraksa, Dr. Britt Hanson, Dr. Carla Martin, Dr. Vanessa McLoughlin, Dr. Rushdie Abuhamdah ,Thomas Goedert, Dr. Carlo Rinaldi, Dr. Yulia Lomonosova, Dr. Catheryn Lim, Katarzyna Chwalenia, Larissa Goli, Dr. Richard Raz, Dr. Ivan Zlatev, Dr. Chandra Vargeese, Dr. Wei Liu, Dr. Susovan Mohapatra, Dr. Chikdu Shivalia, Dr. Jack Godfrey, Dr. Stephany Standley, Nayantara Kothari, and Frank Favaloro.

Having spent many years in academia and industry I owe a lot to the people who shaped me as a person and as a scientist. I am incredibly grateful to have had the opportunity to start my career in the laboratories of Prof. Heidi Tissenbaum and Prof. Marc Vidal, and to continue my development at Alnylam Pharmaceuticals and Ontorii - Wave Life Sciences. During this time, I had the privilege to be mentored by exceptional scientists, and I would like to share this achievement with: Prof. Gregory Verdine, Dr. Stuart Milstein, Dr. Denis Dupuy, Dr. Yosef Landesman, Dr. Rachel Meyers, Dr. Jeff Parker, and Dr. Aaron Morris.

I will be forever indebted to my parents Zoran and Blagica Svrzikapa for their unconditional love, unwavering support, and the values instilled in me and my brother Nebojsa. I am also immensely grateful to my in-laws Snezana and Petar Badnjar, for being wonderful grandparents to our son, and helping us navigate the first year of parenthood.

Finally, I thank my wife Tijana Badnjar for her love, wisdom, dedication, and belief in me, and my son Julian Svrzikapa who was born a few weeks after I started my DPhil studies and has continuously showered good fortune on this work and our lives. They have shared every moment of this incredible journey with me, so it only seems right to dedicate this dissertation to them.

Abstract

Upstream open reading frames (uORFs) are *cis*-regulatory sequence elements characterized by an initiation codon located in the 5′ untranslated region (5′ UTR) and an associated in-frame termination codon located upstream or downstream of the primary ORF (pORF). uORFs have been found to inhibit translational initiation at the pORF by sequestering ribosomes through premature initiation in the 5′ UTR. It was therefore hypothesized that that the disruption of uORFs would result in increased translation at the pORF.

Initially, we predicted occurrences of uORFs in the human transcriptome and generating a database (*uORFeome*) which annotates uORF features. To better understand the complex mechanisms by which uORFs inhibit protein expression, large-scale transcriptomics and proteomics datasets from 32 human tissues were combined with uORF predictions to perform a comprehensive bioinformatics analysis of the influence of uORF features. *uORF Detective* – a transcriptome-wide uORF visualization tool, was developed to enable the study of uORFs in context of ribosome profiling data. A deep learning model was used to predict uORF effects and discover novel sequence features with significant effects on uORF translational repression.

The model predicted that AUG-to-UUG uORF initiation codon changes would result in translational derepression (a well-established uORF property), and that minimal uORFs (an initiation codon immediately followed by a stop codon) would also be inhibitory (an unexpected finding). Furthermore, the presence of a tryptophan residue at position 5 of the uORF-encoded peptide was predicted to enhance uORF-mediated repression. All of these predictions were subsequently confirmed experimentally using reporter assays.

Additional bioinformatics and reporter assay analyses revealed a lack of activity of uORFs proximal to the cap, a discovery of initiation at a minimal uORF in the 5′ UTR of the

ATF4 gene, identification of potentially therapeutically-relevant uORF target sites that may be utilized to upregulate expression of the *BDNF* and *SCN1A* genes, as well as position- and Kozak sequence-dependent effects on uORF activity. This work demonstrates that in cases where initiation at uORFs inhibits productive translation at the pORF, the disruption of the uORF initiation codon can result in increased translation of the pORF. Such interference with regulatory uORFs may be of benefit in multiple medical conditions where an increase in protein levels is desirable (e.g., haploinsufficiency disorders).

Oligonucleotide (ON) targeting strategies were explored to evaluate their ability to interfere with uORF activity. Steric block ONs were utilized to target uORF initiation codons. No upregulation was observed in any experiment using this strategy, suggesting that the previously published findings are not reproducible. As an alternative approach, RNA editing ONs were used to disrupt the uORF initiation codon by recruiting adenosine (A) deaminases acting on RNA (ADAR) enzymes. Sequence-specific RNA editing was observed in a variety of endogenous uORF targets at low efficiencies and in some cases, reporter gene activation was also observed. The importance of sequence context surrounding the target editing site was identified as a factor in successful editing, with C at the 5′ of the target A being particularly unfavorable. The inclusion of inosine (I) nucleotides opposite of C in the in the nearest neighbor position was found to partially recover RNA editing activity. While this editing was observed in context of interferon alpha (IFNα)-induced ADAR overexpression, future chemistry optimization may facilitate the generation of higher potency RNA editing ONs, which can be used to activate gene expression by interfering with uORFs.

Table of Contents

Declaration ..... **I**
Acknowledgements ..... **II**
Abstract ..... **IV**
Table of Contents ..... **VI**
List of Tables ..... **XI**
List of Figures ..... **XII**
List of Abbreviations ..... **XVI**
1 Introduction **23**
1.1 Canonical translation in eukaryotic organisms 23
1.1.1 Eukaryotic Translation: Initiation 23
1.1.1.1 Leaky scanning and the Kozak consensus sequence 27
1.1.1.2 Rate-limiting features of translation initiation 29
1.1.1.3 Non-AUG initiation 31
1.1.1.4 Noncanonical translation initiation 35
1.1.2 Eukaryotic Translation: Elongation 38
1.1.2.1 Rate-limiting features of translation elongation 41
1.1.3 Eukaryotic Translation: Termination 41
1.1.4 Eukaryotic Translation: Ribosome recycling 43
1.2 Post-transcriptional regulation and the correlation between the human transcriptome and the proteome 44
1.2.1 Post-transcriptional regulation by *trans*-acting elements 45
1.2.2 Post-transcriptional regulation by *cis*-acting elements 46
1.3 Upstream Open Reading Frames (uORFs) 48
1.3.1 Ribosome uORF interactions 50
1.3.2 Translation reinitiation after uORF translation 52
1.3.2.1 Canonical initiation factor-mediated reinitiation 53
1.3.2.2 Noncanonical reinitiation factor-mediated reinitiation 54
1.3.2.3 Reinitiation by post-termination migrating 80S ribosomes 55
1.3.3 uORF-dependent translation reinitiation at the pORF 56
1.3.4 uORFs may promote mRNA degradation 58
1.3.5 Are uORF-encoded microproteins functional? 59
1.3.6 uORF implication in human disease 60
1.3.7 Therapeutic potential of targeting uORFs 62

1.4 Oligonucleotide therapeutics 62
1.4.1 Chemical modifications of oligonucleotide therapeutics 63
1.4.1.1 Modifications of the phosphodiester bond 64
1.4.1.2 Base modifications 65
1.4.1.3 2′ Modifications 66
1.4.1.4 ON delivery conjugates 68
1.4.2 Modes of oligonucleotide target engagement 69
1.4.2.1 RNase H-recruiting gapmer ASOs 69
1.4.2.2 RNA interference (RNAi) 70
1.4.2.3 Splice-switching steric blocking ONs 71
1.4.2.4 RNA editing 73
1.4.3 Challenges in oligonucleotide drug development 74
1.4.3.1 Innate immune defense mechanisms 75
1.4.3.2 Chemical modification architectures 76
1.4.3.3 Internalization, size, and charge 76
1.4.3.4 Off-target effects 77
1.4.3.5 Targeted tissue delivery 78
1.4.4 Oligonucleotides as personal medicines 78
Thesis hypotheses and aims 80
2 Materials and methods 81
2.1 Computational analyses 81
2.2 Cell culture 81
2.3 RNA extraction 81
2.4 cDNA synthesis 82
2.5 Reporter assay vector generation 82
2.5.1 Generation of the psiCheck dual-luciferase reporter (pDLR) 5′ UTR vector 82
2.5.2 Reporter assay insert design and generation of the pDLR GOI 5′ UTR plasmids 83
2.5.3 Generation of the pDLR HOXA11 5′ UTR uORF vector 84
2.6 Transfection of Dual-Glo luciferase 5′ UTR model plasmids 24 hours 84
2.7 Dual-Glo luciferase (DLR) assay 85
3 Results 1: Bioinformatics analysis of uORFs 86
3.1 Introduction 86
3.2 Hypothesis and Aims 88
3.3 Materials and Methods 89
3.3.1 Prediction and mapping of uORFs in the human transcriptome (uORFeome) 89
3.3.2 Processing of uORF feature sets 89
3.3.3 Processing of the proteomics data 90

3.3.4 A strategy for generating unique genomic uORF identifiers 90
3.4 Results 93
3.4.1 uORFeome a database of predicted uORFs and their properties in the human transcriptome 93
3.4.2 Probing the effects of uORFs on translation in matched transcriptomics and proteomics datasets 97
3.4.2.1 uORFs are less represented in highly expressed human transcripts across tissues 98
3.4.2.2 Filtered genes are not enriched in uORFs 100
3.4.2.3 Gene Ontology (GO) term enrichment analysis of the highly expressed transcript population 102
3.4.3 Bioinformatics analysis of uORF features and their effects on protein translation 102
3.4.4 uORF detective – a novel tool to power the study of uORFs 107
3.4.4.1 A tool for mapping Ribo-Seq data to the genome (GWIPS) 107
3.4.4.2 Development and features of the uORF detective software 109
3.4.4.3 ATF4 and uORF-mediated differential regulation of translation during stress conditions 112
3.4.4.4 Validation of a new translational model for the *ATF4* gene 118
3.4.5 Predicting the repressive potential of uORFs 122
3.4.5.1 Predicting the regulatory effects of 5′ UTRs from sequence 123
3.4.6 Probing the Optimus 5-Prime model for novel uORF features 125
3.4.6.1 Generating MRL predictions for two million randomly generated uORFs 125
3.4.6.2 AUG -> UUG mutation of the initiation codon is predicted to result in increased MRL 128
3.4.6.3 Common feature analysis of the tails of the distribution reveals novel motifs 129
3.4.6.4 Investigating amino acid influence in the most inhibitory sequence 130
3.4.6.5 Validation of the predicted motifs using the pDLR HOXA11 5′ UTR uORF system 131
3.4.6.6 Machine learning modeling of the effects of minimal uORFs with Optimus 5-Prime 133
3.5 Discussion 135
3.5.1 uORFeome 136
3.5.2 Limitations of ribosome profiling studies 136
3.5.3 uORF Detective 137
3.5.4 Minimal uORFs 138
3.5.5 Machine learning – potential and limitations 139
3.5.6 Future prospects of computational uORF analyses 140
3.5.7 Conclusions 141
4 Results 2: Modeling the repressive potential of uORFs in disease-relevant genes 143
4.1 Introduction 143
4.2 Hypothesis and aims 145

4.3 Materials and methods 146
4.3.1 Selection of an oncogene and tumor suppressor gene panel 146
4.3.2 Deletion walk strategy 146
4.4 Results 148
4.4.1 Translational upregulation of the pORF by weakening uORF initiation codon in a panel of oncogenes and tumor suppressor genes 148
4.4.2 Translational upregulation of the pORF by weakening uORF initiation codon in neuromuscular disease 152
4.4.3 Mechanism of translation repression in genes with multiple uORFs 154
4.4.4 A deletion walk for investigating the presence of inhibitory elements in the 5′ UTRs of therapeutically-relevant GOIs which lack uORFs 162
4.4.5 Investigating the effect of uORF position on translation at the pORF 164
4.5 Discussion 167
4.5.1 uORF as potential targets for neuromuscular diseases and cancer 167
4.5.2 uORF features and their contribution towards uORF activity 169
4.5.3 Future prospects of uORF applications to therapeutic development 170
4.5.3.1 Towards more predictive ribosome profiling datasets 170
4.5.3.2 Harnessing uORFs for targeted gene regulation 171
4.5.4 Conclusions 172
5 Results 3: Targeting uORFs with steric block and RNA editing oligonucleotide technologies 174
5.1 Introduction 174
5.2 Hypothesis and aims 178
5.3 Materials and methods 179
5.3.1 Transfection of editing antisense oligonucleotides in the presence of human interferon alpha A (IFNα) 179
5.3.2 Cotransfection of Dual-Glo luciferase 5′ UTR model plasmids and steric block ASOs with Lipofectamine 2000 179
5.3.3 Dual transfection of pDLR GOI 5′ UTR plasmid DNA and steric block ASOs with Lipofectamine 3000 and RNAiMAX 180
5.3.4 Co-transfection of ADAR p110 and 5′ UTR model plasmids, and sequential transfection of editing antisense oligonucleotides in the presence of human interferon alpha A (IFNα) 180
5.3.5 *RNASEH1* steric block ASO reverse transfection of HeLa cells with RNAiMAX for protein harvesting in 24-well plates 181
5.3.6 Quantitative capillary protein analysis 181
5.3.7 Sanger Sequencing PCR 183
5.3.8 Automated gel electrophoresis and quality control of uORF PCR 186
5.3.9 Enzymatic PCR product purification 186

5.3.10 Sanger sequencing premix reaction 187
5.3.11 Analysis of Sanger sequencing trace data 187
5.3.12 Formulation of double-stranded (DS) ADAR-recruiting ON designs 188
5.4 Results 189
5.4.1 Targeting uORFs with steric block ONs 189
5.4.1.1 Steric block ASOs targeting uORF initiation codons are not active promoters of upregulation 189
5.4.2 Targeting uORFs with RNA editing ONs 195
5.4.2.1 RNA editing with double-stranded (DS) ONs 197
Interferon alpha induced ADAR expression in HeLa, SF8628 and HEK293T 200
5.4.2.2 Endogenous A-to-I RNA editing with stem loop ONs 202
5.4.2.3 Endogenous A-to-I RNA editing with AIMers 204
5.4.2.4 DNA Inosine placement opposite of a C nearest neighbor improves editing 207
5.4.2.5 A-to-I RNA editing of uORF initiation codons for the purpose of protein upregulation 209
5.5 Discussion 211
5.5.1 The problem with steric block ASOs in the 5′ UTR 211
5.5.2 RNA editing of uORFs 212
5.5.3 Future prospects of uORF ON research 214
5.5.4 Conclusions 215
6 Discussion and conclusions 217
7 References 225
8 Appendix 247

List of Tables

Table 1. Classification of viral IRESs 37

Table 2. FDA-approved siRNA therapeutics featuring a triantennary GalNAc moiety 68

Table 3. Primers used to amplify regions containing editing targets for Sanger sequencing.....185

Table 4. PCR cycling condition for Sanger sequencing 186

List of Figures

Figure 1. A visual representation of a translating ribosome 24

Figure 2. Assembly of the 80S initiation complex post eIF1 dissociation....27

Figure 3. Translation elongation 39

Figure 4. Translation termination.....42

Figure 5. Upstream Open Reading Frames (uORFs).....49

Figure 6. Ribosome uORF interactions and their effect on translation 51

Figure 7. Typical locations of oligonucleotide chemical modifications 64

Figure 8. Common ON modifications at the 2′ position of the RNA ribose. 66

Figure 9. Design of the pDLR 5′ UTR vector and GOI 5′ UTR cloning strategy. 83

Figure 10. Strategy for generating unique genomic uORF identifiers 92

Figure 11. Bioinformatics analysis of uORFs properties in human and mouse.....94

Figure 12. A current uORF analysis on the uORFeome database (Analyzed May 2022).....96

Figure 13. Heatmap of uORF frequency across mRNA expression levels..99

Figure 14. Heatmap of uORF frequency across mRNA expression levels (Unfiltered). 101

Figure 15. CDF plots of the protein expression of genes that contain or do not contain a uORF.....103

Figure 16. Cumulative probability plots of uORF properties with *P*-values computed using the Kolmogorov-Smirnov test.....106

Figure 17. Ribosome profiling and uORF data for the *HOXA11* and *TP53* genes visualized in GWIPS.....108

Figure 18. Algorithm process diagram for uORF Detective. The software generates an interactive HTML and a JavaScript data dependency file. .110

Figure 19. Ribosome profiling data of HOXA11 visualized in uORF Detective. 112

Figure 20. Published model of *ATF4* translation during normal and stress conditions. 114

Figure 21. uORF detective analysis of the *ATF4* gene 116

Figure 22. A proposed translation model for the *ATF4* gene. 117

Figure 23. Modeling the uORF regulation of the *ATF4* gene.....119

Figure 24. A proposed model for *ATF4* translational regulation 121

Figure 25. Machine learning predictions of poly-E repression 124

Figure 26. MRL predictions on simulated uORFs in the HOXA11 5′context 126

Figure 27. Sequence logo representing the amino acids frequency of the 100 most inhibitory internal Methionine-containing sequences.....127

Figure 28. XY plots of the predicted MRL of two million sampled AUG-initiated uORFs and their UUG-initiation mutated analogs 128

Figure 29. The regulatory potential of uORFs may be influenced by *cis*-elements in the uORF-encoded sequence.....130

Figure 30. *In silico* Alanine scan to evaluate individual amino acid contributions..... 131

Figure 31. Validation of machine learning predicted MRLs.....133

Figure 32. Predicting the effects of introducing a minimal uORF on MRL 134

Figure 33. Deletion walk strategy for the detection of non-uORF inhibitory elements in the 5′ UTR 147

Figure 34. Evaluation of the regulatory potential of 5′ UTRs containing wild-type or mutated uORFs in a Dual-Glo Luciferase reporter assay 150

Figure 35. Evaluation of the sensitivity of the Dual-Glo Luciferase reporter assay.....151

Figure 36. Evaluation of the regulatory potential of *bona fide* 5′ UTRs containing wild-type or mutated uORFs in a Dual-Glo Luciferase reporter assay..... 152

Figure 37. Evaluation of the regulatory potential of *bona fide* 5′ UTRs containing wild-type or mutated uORFs in neuromuscular disease GOI.154

Figure 38. Evaluation of the regulatory potential of individual uORFs in the *BRD3*.....156

Figure 39. Dual luciferase assay evaluating the individual and combinatorial effects of uORFs in the multi-uORF 5′ UTRs of two splice variants of the *BDNF* gene. 158

Figure 40. Dual luciferase assay evaluating the individual and combinatorial effects of uORFs in the multi-uORF 5′ UTRs of two splice variants of the *SCN1A* gene. 161

Figure 41. A deletion walk strategy to search for inhibitory elements in the 5′ UTR in disease relevant GOIs 163

Figure 42. uORF activity is dependent on position and initiation context166

Figure 43. Optimization of Jess conditions for RNase H1 and Vinculin antibodies using HeLa lysate.....183

Figure 44. Example of RNA base editing in the 3′ UTR of *GAPDH* analyzed by EditR 188

Figure 45. uORF targeting ONs downregulate expression at the RLuc pORF (relative to control ASO treatment).....191

Figure 46. uORF targeting ONs downregulate expression at the RLuc pORF (relative to mock treated wild type vector).....192

Figure 47. Dual transfection plasmid and 2' O-Methyl modified ON targeting the uORFs of *RNASEH1* and *HOXA11*.....193

Figure 48. RNase H1 protein levels after treatment with steric block ONs targeting the uORF of *RNASEH1* in HeLa cells.....194

Figure 49. Oligonucleotide designs used to target uORF initiation codons 197

Figure 50. Transfection of double-stranded (DS) RNA editing ON designs 199

Figure 51. The observed upregulation effect of the 2′-MOE DS ONs is not due to sequence-specific RNA editing 200

Figure 52. Gene expression analysis of IFNα induced ADAR expression in HeLa, SF8628 and HEK293T cells 201

Figure 53. Endogenous A-to-I RNA editing using stem loop (SL) ADAR-recruiting ONs 203

Figure 54. Endogenous A-to-I RNA editing of a target adenosine in the 3′ UTR of the *GAPDH* gene with stem loop ONs is dependent on levels of induced ADAR.....204

Figure 55. ON chemistry of used AIMers as described by Monian *et al.*..205

Figure 56. Optimal A-to-I RNA editing with AIMer ONs can be efficacious even without IFNα ADAR induction, while uORF initiation codon editing may be more challenging.....206

Figure 57. uORF initiation codons can be edited with ON technologies...208

Figure 58. A-to-G mutations in uORF initiation codons silence uORF activity.....209

Figure 59. Dual transfection of constructs carrying *bona fide* 5′ UTRs and AIMers in the presence of IFNα show both upregulation and downregulation of RLuc pORF translation 210

List of Abbreviations

2′-OMe	2′-O-methyl
2′-MOE	2′-O-methoxyethyl
3′ UTR	Three prime untranslated region
40S	Small ribosomal subunit
5meC	5-Methylcytosine
5′ UTR	Five prime untranslated region
60S	Large ribosomal subunit
A-site	Aminoacyl site
A	Adenosine
ADAR	Adenosine deaminase acting on RNA
adRNA	ADAR RNA
Ago2	Argonaute 2
Aimer	A-to-I oligonucleotides
ALS	Amyotrophic lateral sclerosis
ASGPR	Asialoglycoprotein receptor
ASO	Antisense oligonucleotide
ATP	Adenosine triphosphate
BED	Browser extensible data format
BED12	A 12 column browser extensible data format
BIGWIG	A variation of the wiggle format
BMD	Becker muscular dystrophy
bp	Base pair
C	Cytosine

cadRNA	Circular ADAR RNA
cDNA	Complementary deoxyribonucleic acid
CDA	Cytidine deaminase
CDF	Cumulative distribution function
CDS	Coding sequence
cGAS	Cyclic GMP-AMP synthase
CHX	Cycloheximide
ClinGen	Clinical Genome
CLR	C-type lectin
CNS	Central nervous system
CPP	Cell-penetrating peptide
CrPV	Cricket paralysis virus
D	Aspartic acid
DNA	Deoxyribonucleic acid
DLR	Dual-Glo luciferase
DM1	Myotonic dystrophy type 1
DM2	Myotonic dystrophy type 2
DMD	Duchenne muscular dystrophy
DMEM	Dulbecco’s modified eagle medium
DPR	Dipeptide repeat protein
ds	Double-stranded
E-site	Exit site
eGFP	Enhanced green fluorescent protein
eIF	Eukaryotic initiation factor
EJC	Exon-junction complex

eEF	Eukaryotic elongation factor
ER	Endoplasmic reticulum
eRF	Eukaryotic release factor
eRNA	Enhancer RNA
EURORDIS	European organization for rare diseases
FLOSS	Fragment length organization similarity score
FLuc	Firefly luciferase
FPKM	Fragments per kilobase of transcript per million mapped reads
FTD	Frontotemporal dementia
FXTAS	Fragile X tremor ataxia syndrome
G	Guanine
GA	Gibson assembly
GalNAc	N-acetylgalactosamine
GAP	GTPase-activating protein
GDP	Guanosine diphosphate
GEF	Guanine nucleotide exchange factor
GO	Gene ontology
GOI	Gene of interest
GTI-seq	Global translation initiation sequencing
GTP	Guanosine triphosphate
GWIPS	Genome wide information on protein synthesis
HCC	Human hepatocellular carcinoma
HCM	Hypertrophic cardiomyopathy
HD	Huntington’s disease
hg38	Human genome version 38

HGMD	Human gene mutation database
HI	Haploinsufficient
HT	Harringtonine
IFNα	Interferon alpha
I	Inosine
IF	Initiation factor
IRES	Internal ribosomal entry site
ISR	Integrated stress reponse
ITAF	IRES-transacting factor
kDa	Kilodaltons
KDE	Kernel density estimation
KS	Kolmogorov-Smirnov
LC-MS	Liquid chromatography – mass spectrometry
LEAPER	Leveraging endogenous ADAR for programmable editing of RNA
LNA	Locked nucleic acid
LNP	Lipid nanoparticle
LTM	Lactimidomycin
LOF	Loss of function
M	Methionine
Met-tRNA$_{iMet}$	Aminoacylated initiator methionine tRNA
miRNA	MicroRNA
miRNP	Micro-ribonucleoprotein
MRL	Mean ribosome load
mRNA	Messenger RNA
MS	Multiple sclerosis

MU	Mutant
NAFLD	Non-alcoholic fatty liver disease
NAT	Natural antisense transcript
NC	Non-canonical
NCC	Near cognate codon
NGS	Next generation sequencing / Second generation sequencing
NMD	Nonsense-mediated decay
NMD	Neuromuscular diseases
NLR	Nucleotide-oligomerization domain protein-like receptor
nt	Nucleotide
OMIM	Online Mendelian inheritance in man
ON	Oligonucleotide
ORF	Open reading frame
P-site	Peptidyl site
paRNA	Promoter-associated RNA
PCR	Polymerase chain reaction
PDA	Phosphorodiamidate
pDLR	psiCheck dual-luciferase reporter
Pen/Strep	Penicillin/Streptomycin
PIC	Preinitiation complex
PK/PD	Pharmacokinetic/Pharmacodynamic
PMO	Phosphorodiamidate morpholino
PN	Phosphoryl guanidine-based backbone
PO	Phosphodiester
Poly-E	Glutamic acid repeats

pORF	Primary open reading frame
PPMO	peptide-PMO
pre-miRNA	Precursor miRNA
Proteoform	A specific protein translation product arising from the same gene
PS	Phosphorothioate
PTC	Premature stop codon
qRT-PCR	Quantitative reverse transcription polymerase chain reaction
R	Arginine
RAN	Repeat-associated non-AUG translation
RBP	RNA binding protein
RESTORE	Recruiting endogenous ADAR to specific transcripts for oligonucleotide-mediated editing
Ribo-Seq	Ribosome profiling
RISC	RNA-induced silencing complex
RLuc	Renilla luciferase
RNA	Ribonucleic acid
RNAa	RNA activation
RNAi	RNA interference
r_s	Spearman rank coefficient
RPF	Ribosome protected fragment
rRNA	Ribosomal RNA
RT-qPCR	Reverse transcription quantitative polymerase chain reaction
RYG	Arginine-Tyrosine-Glycine
saRNA	Small activating RNA
SCA8	Spinocerebellar ataxia type 8

siRNA	Small interfering RNA
SL	Stem loop
SMA	Spinal muscular atrophy
SNP	Single nucleotide polymorphism
SRL	Sarcin-ricin loop
STING	Stimulator of interferon genes
SV40	Simian virus 40
TANGO	Targeted augmentation of nuclear gene output
TIS	Transcript initiation site
TC	Ternary complex
*T*m	Melting temperature
TLR	Toll-like receptor
TMT	Tandem mass tag
TPM	Transcripts per kilobase million
TRBP	Transactivation response element RNA-binding protein
tRNA	Transfer RNA
TSG	Tumor suppressor gene
U	Uracil
uAUG	Upstream AUG translation initiation codon
uORF	Upstream open reading frame
W	Tryptophan
WFS2	Wolfram syndrome 2
WIG	Wiggle format
WT	Wild type

1 Introduction

1.1 Canonical translation in eukaryotic organisms

Translation of messenger RNA (mRNA) is a conserved, highly regulated, and dynamic process represented by four distinct stages: initiation, elongation, termination, and ribosome recycling; each involving specific sets of factors and conditions[1].

1.1.1 Eukaryotic Translation: Initiation

Translation initiation, the first stage of protein synthesis, is a complex process of ribosome recruitment and assembly at translation initiation sequence signals. Initiation can be thought of as a critical rate-limiting process and has been shown to be a major determinant of the overall rate of translation in both prokaryotes and eukaryotes[2-4]. This notion is also supported by the observed accumulation of ribosomes at initiation sites in ribosome profiling studies[5].

Initiation employs at least 12 eukaryotic initiation factors (eIFs) that drive the recognition of a favorable sequence to initiate translation, and the subsequent assembly of elongation-competent 80S ribosomes. In eukaryotes, 80S ribosomes are complex protein-rRNA (ribosomal RNA) structural assemblies comprised of two major subunits: the small ribosomal subunit (40S) – whose function is to scan and decode genetic information encoded by the mRNA, and the large ribosomal subunit (60S) – which joins with the 40S to catalyze peptide bond formation[6]. The decoding process involves the interaction between the mRNA, ribosome, and transfer RNAs (tRNAs) – ribonucleoprotein molecules which deliver corresponding amino acids to the ribosome. Depending on whether they carry an amino acid, peptide, or no group at the 3′, tRNAs can exist in three different states: (i) aminoacyl-tRNA, (ii) peptidyl-tRNA, and (iii) deacylated tRNA (**Figure 1**). There are three specific sites on the

ribosome where tRNAs bind during translation: the aminoacyl site (A-site) which accepts an incoming aminoacyl-tRNA, the peptidyl site (P-site) which holds the tRNA with the nascent peptide chain, and the exit site (E-site), which is where the deacylated tRNA is held before it exits the ribosome (**Figure 1**) [7].

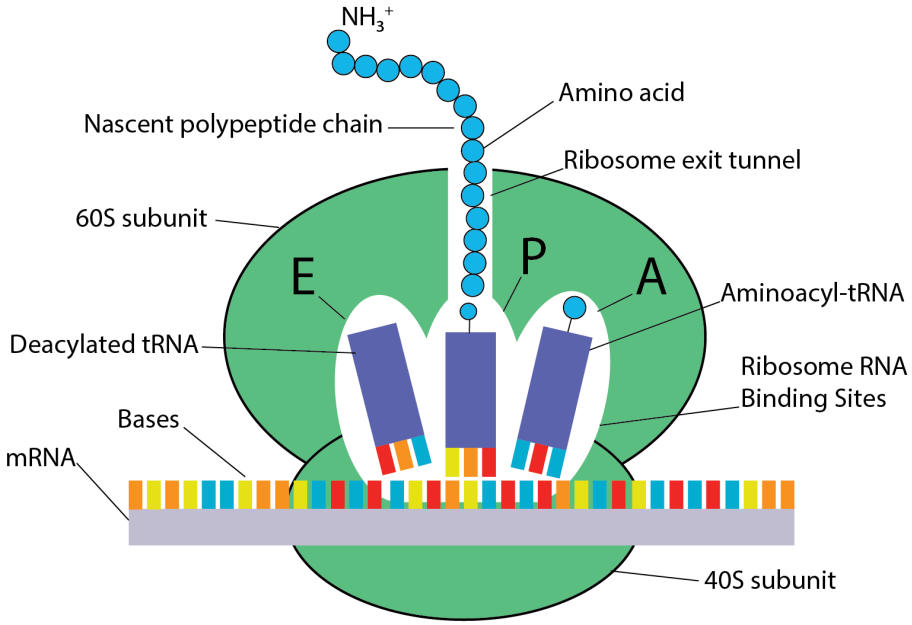


Figure 1. A visual representation of a translating ribosome

The 80S ribosome is assembled by joining the 40S and 60S ribosomal subunits. Aminoacyl-tRNA carrying an amino acid enters from the A-site. The amino acid nascent peptide chain exits through the ribosome exit tunnel at the P-site. Deacylated tRNA exits through the E-site.

Translation initiation starts with the recruitment of the heterotrimeric eukaryotic initiation factor 4F (eIF4F) protein complex to the 7-methylguanosine cap at the 5′ terminus of mRNA transcripts. eIF4F is comprised of three structural subunits: the cap-binding protein eIF4E, the DEAD-box RNA helicase eIF4A, and the large scaffold protein eIF4G[8]. Similar to eIF4F, many of the eukaryotic initiation factors (eIFs), the proteins that facilitate the process of translation, are heteromeric protein complexes[6].

eIF4G binds to the poly(A)-binding protein (PABP), which facilitates mRNA circularization. It has been shown that the PABP/poly-(A) interaction stabilizes the poly(A) tail, and consequently protects the mRNA from endonuclease degradation[9,10]. In addition to mRNA stabilization, the PABP/eIF4G interaction has been implicated in stimulating translation via: (i) Promotion of ribosome recycling – by bringing the ribosome in proximity to the cap; (ii) Acting as an initiating factor by stimulating 60S ribosome joining; and (iii) increasing the affinity of the eIF4F for the cap – resulting in enhanced 40S ribosome recruitment[4].

Secondary structures near the cap are unwound by the helicase action of eIF4A and eIF4F; an activity significantly enhanced by the dimer eIF4B. More generally, RNA helicases catalyze the ATP-dependent processes of mRNA secondary structure rearrangement, RNA/RNA interaction disruption, and rearrangement of RNA/protein complexes in multiple RNA-related events including transcription, editing, splicing, RNA export, ribosome biogenesis, RNA/protein assembly, translation, and RNA degradation. However, in the context of translation initiation, helicase activity near the cap structure is thought to yield a single-stranded mRNA region optimal for the recruitment of the 40S ribosomal subunit by the eIF4G-bound eIF3 – the largest and most complex eIF comprised of 13 different subunits[11-13]. eIF3 tethers the 40S ribosomal subunit to eIF4F, but also serves to prevent premature 60S ribosome assembly[13,14]. When the 40S ribosomal subunit is bound to eIF3, eIF1, eIF1A, and eIF5, it promotes the recruitment of the Met-tRNA$_i^{Met}$•eIF2•GTP ternary complex (TC) and formation of the 43S preinitiation complex (PIC)[15,16]. The TC serves to deliver the aminoacylated initiator methionine tRNA (Met-tRNA$_i^{Met}$) to the 40S subunit, to participate in initiation codon recognition and to encode the first amino acid of the protein. The assembled 43S PIC, where Met-tRNA$_i^{Met}$ is loosely positioned in the peptidyl site (P-site), binds to the 5′ cap in tandem with eIF4F to create the 48S translational initiation complex. The 48S initiation complex then

migrates base by base, scanning the mRNA sequence until it encounters an AUG codon, ideally residing in an optimal sequence context (discussed in detail in the following section). This scanning is facilitated by eIF1 and eIF1A, which promote an open conformation of the 48S initiation complex and discriminate against codon mismatches. The open conformation of the 48S initiation complex is maintained by the C-terminal tail of eIF1A, which is proposed to prevent Met-tRNA$_i^{Met}$ from being fully engaged with the P-site[17-19]. Namely, during scanning, the GTPase-activating protein (GAP) eIF5 promotes the irreversible hydrolysis of eIF2•GTP. However, up until an initiation codon is reached the phosphate P_i remains engaged with eIF1; which prevents its premature release[20-22]. The GTP hydrolysis of eIF2•GTP was long thought to occur at the AUG initiation codon[23-25]. However, more recent studies suggest that it is the release of the inorganic phosphate P_i, and not the GTP hydrolysis itself that is controlled by the recognition of the AUG codon[23].

When a sequence context favorable for initiating translation is reached, Met-tRNA$_i^{Met}$ becomes fully engaged in the P-site, and the 48S initiation complex transitions from an open to a closed conformation. This triggers the release of eIF1, which results in dissociation of the phosphate P_i from the hydrolyzed eIF2•GTP[7,20,26-28]. The assembly of the 80S ribosome is blocked by the presence of eIF1, eIF1A, eIF3 and residual eIF2•GDP. After the dissociation of eIF1, eIF2, eIF3, eIF5, eIF4A, and eIF4B the joining of the 40S and 60S ribosomal subunits is blocked by eIF1A which remains engaged at the A-site (**Figure 2a**). eIF1A recruits the 60S subunit through its interaction with the ribosome-dependent GTPase eIF5B. During this interaction, eIF5B together with the 60S ribosomal subunit displaces eIF2•GDP. eIF5B then interacts with the C-terminal tail of eIF1A, displaces eIF1A, and upon hydrolysis of its bound GTP co-dissociates together with eIF1A to allow for 80S ribosome assembly (**Figure 2b**)[7]. This results in a fully assembled 80S initiation complex, where the aminoacyl-tRNA carrying

Methionine is located at the P site, the E-site is empty, and the A-site is empty and primed to accept the next aminoacyl-tRNA (**Figure 2c**).

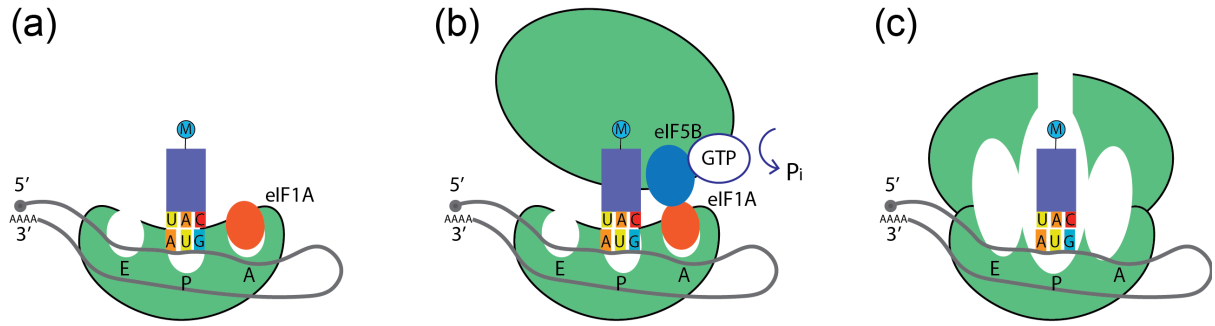


Figure 2. Assembly of the 80S initiation complex post eIF1 dissociation

A diagram showing the process of ribosome assembly after the dissociation of eIF1. (a) After the release of eIF1, eIF2, eIF3, eIF5, eIF4A and eIF4B the only initiation factor engaged with the 40S subunit is eIF1A which is present at the A-site, the initiator tRNA carrying Methionine is at the P-site, the E-site is empty, while the mRNA is circularized where the cap and poly-A tail interaction is mediated by PABP (b) In the next stage the 60S-bound eIF5B-GTP interacts with the eIF1A at the A-site. (c). The GTP hydrolysis results in the release of both eIF5B and eIF1A resulting in a complete assembly of the 80S initiation complex, where the A-site can accept the next aminoacyl-tRNA.

1.1.1.1 Leaky scanning and the Kozak consensus sequence

Evidence for the mechanism by which the scanning 48S initiation complex finds the AUG codon, known broadly as the scanning mechanism of translation initiation, emerged as early as 1978, after Kozak observed unusual polysome-like complexes when the AUG codon recognition blocker edeine was introduced to cells. Kozak and Shatkin observed that in the presence of edeine the 40S ribosomal subunits are capable of moving across the mRNA sequence[29]. Subsequent studies have demonstrated that the ribosome advances linearly, and scans for an initiation codon. This linear scanning to the initiation codon, rather than skipping, has been further demonstrated by the observation that introduction of base-pairing structures between the 5′ cap and the first AUG codon represses translation initiation[30].

Subsequent studies by Kozak in the 1980S, building on her work describing the scanning mechanism of translation, revealed that sequence context is critical for the recognition of the AUG codon as an initiation start. In a sequence analysis of 153 mRNAs, it was revealed that the sequence context flanking the AUG codons is non-random[31]. It was initially found that the sequence RNNAUGG (where N can be any base, and R is a purine) is a consensus sequence surrounding the initiation codon. Furthermore, it became evident that positions -3 and +4 relative to the AUG codon are highly conserved, and likely critical, with 151/153 transcripts having either a purine at position -3, or a G at position 4, or both. In two subsequent studies with larger sets of cellular mRNAs, the consensus sequence was updated to CCRCCAUGG, and later expanded to (GCC)GCCRCCAUGG (where the bases in parentheses are of uncertain significance). These studies revealed that 97% of initiation codon context sequences have a purine, and most commonly an A (61%) at the -3 position[32,33]. Site-directed mutagenesis *in vitro* studies further established the contribution of the -3 and +4 positions as critical for translation initiation, with initiation efficiency varying 5 to10-fold when either position was substituted, and up to 20-fold when both purines were substituted[31,34-36]. Kozak also observed significant fluctuations in relative proinsulin yields with base changes at position -3 and +4 relative to the first position of the AUG codon *in vivo*[37]. The Kozak consensus motif has been largely corroborated by multiple studies with minor modifications and minor species-specific preferences in vertebrates[38-41]. The consensus motif in vertebrates is now generally accepted to be GCMRNCAUGG (Where M is either A or C)[41].

Two studies established a critical link between the ability of the 40S ribosome to scan for the initiation codon and translation initiation. Namely, it was shown that alterations at the -3R and/or +4G critical Kozak motif sites promotes initiation codon readthrough through a mechanism known as "Leaky scanning"[34,35,42,43]. It is now well-established that the Kozak motif is a critical factor influencing the probability of initiation, and consequently an important

cis-regulatory element of translation. Based on their initiation potential, guided by the presence of specific bases at the most critical positions, Kozak motifs can be classified as: "optimal" – GCCRCCAUGG, "strong" – NNNRNNAUGG, "adequate" – NNNRNNAUG(A/C/U) or NNN(C/U)NNAUGG, or "weak" – NNN(C/U)AUG(A/C/U)[44].

Conservation of the Kozak consensus sequence is critical for translation initiation. Therefore, it can be proposed that mutations that disrupt the Kozak motif may have a repressive or activating effect on the open reading frame (ORF) translation. Early work on α-thalassemia has revealed that disruptions of the Kozak sequence can result in pathogenic downregulation of the primary ORF (pORF) – the annotated functional primary proteoform of the gene. It has been reported that a change in position -3 in the α-globin initiation sequence context from CCCACCAUG to CCCCAUG, caused by a 2 nucleotide AC deletion, resulted in a decrease of globin translation, and was hypothesized to be the cause for α-thalassemia[45]. A subsequent study has revealed that this alteration in the Kozak motif reduced translation efficiency of synthetic α-globin mRNA by 35-50%[46]. In the past three decades, several other conditions have been associated with mutations in the Kozak context sequence including: Graves' disease (*CD40*), coronary artery disease and myocardial infarction (*GP1BA*), Ataxia with vitamin E deficiency (α-*TTP*), and breast/ovarian cancers (*BRCA1*)[47-50].

1.1.1.2 Rate-limiting features of translation initiation

Translation initiation is a rate-limiting and highly regulated process in protein synthesis[51,52]. Within the complex phase of initiation, three key processes can be highlighted as critical for the rate of initiation.

The first is the availability of eIF4E, which is dependent on the phosphorylation of eIF4E-binding protein (4E-BP) by the mTOR kinase. 4E-BP can bind eIF4E and inhibit the recruitment of the 40S subunit to the cap structure[53]. If 4E-BP is not phosphorylated, the

eIF4E/eIF4G interaction is competitively inhibited by 4E-BPs, which prevents the formation of the eIF4F complex, and its subsequent assembly at the cap structure[54].

The second is the availability of ternary complex, which depends on the availability of eIF2 and recycling of eIF2•GDP. Under normal conditions, eIF2 is recycled by being recharged with GTP by eIF2B – a guanine nucleotide exchange factor (GEF). However, under specific cellular stress conditions, a serine 51 phosphorylation of the eIF2α subunit by several possible kinases, turns eIF2 into a competitive inhibitor of eIF2B, which prevents eIF2 recycling, decreases the levels of the available TC, and ultimately results in repression of global translation[26,55]. The acquisition of ternary complex is particularly important for sequences with multiple initiation codons in their 5′ UTRs, as it is required to recycle and recharge scanning ribosomes to make them competent for initiation at the pORF[1].

The third critical feature that has significant impact on the proteome is the Kozak sequence – which in agreement with the leaky scanning model, is a variable that influences the probability with which a scanning ribosome can recognize the initiation codon and commence elongation. An unfavorable Kozak consensus sequence promotes leaky scanning and initiation codon bypass, which is predicted to result in decreased initiation and reduced protein levels[35]. Interestingly, most initiation codons in the human transcriptome, including that of the pORF, do not feature a strong Kozak consensus sequence. Recent studies in *Drosophila* have observed the same phenomenon and shown that the protein levels of transcripts with suboptimal Kozak sequences are less sensitive to drops in global elongation rates, and more sensitive to drops in initiation rates. In addition to this, Gene Ontology enrichment analysis in the same study revealed an enrichment of weak Kozak transcript in genes involved with transcriptional regulation and the nervous system[56]. These findings suggest that the Kozak consensus sequence, may be a *cis* regulatory feature used for a global co-regulation of specific sets of genes, under various conditions. The consensus sequence around initiation sites may

have significant downstream effects on the levels of protein translation, as the difference in initiation efficiencies between weak and strong Kozak consensus sequences in naturally occurring AUG initiation sites has been estimated to be as much as 12-fold[57].

1.1.1.3 Non-AUG initiation

Eukaryotic translation initiation relies on the recognition of the AUG initiation codon, and the Watson-Crick interaction between AUG and Met-tRNA$_i$ in the ribosome P-site. While the codon-anticodon interaction of the first two positions is strictly regulated at the aminoacyl-acceptor site (A-site) – where aminoacyl tRNA is recruited, base pairing at the P-site may be less stringent. Given that during elongation codon matching takes place at the A-site, while initiation codon recognition occurs at the P-site, the reduced control for base-pairing at the P-site may enable Met-tRNA$_i$ to initiate at codons other than AUG[58]. The ability for non-AUG near-cognate codons (NCCs) to initiate and generate full-length mRNA has been shown in early *in vitro* reporter assays, mass spectrometry studies, and assays that enable positional mapping of ribosomes on the transcriptome[59-61]. Global translation initiation sequencing studies (GTI-seq) performed on mammalian HEK293 cells reveal that while the majority (51%) of transcript initiation sites (TIS) utilize AUG as a translation initiator, a significant proportion (40%) can be attributed to NCC initiation. NCC initiating TIS were found to be particularly enriched for the CUG codon (16%)[62]. Interestingly, it has been shown that prokaryotic NCC TIS which account for ~20% of TIS, tend to utilize GUG and UUG codons[63].

All putative AUG and NCC initiation codons in the 5′ UTRs can be computationally annotated. However, beyond the mapping of AUG and NCCs motifs, without experimental evidence it is difficult to know which one of these positions are *bona fide* initiation sites. To better understand the translational control of *cis*-acting elements that promote initiation, the development of techniques which can measure the rate of initiation at these sites is desirable.

One such technique is ribosome profiling, which coupled with next generation sequencing (NGS) is a powerful tool for mapping the location of ribosome-protected fragments (RPFs) on a transcriptome scale. Conceptually, the standard technique of ribosome profiling relies on fixing ribosomes in place by treatment with cycloheximide (CHX) – a commonly used protein synthesis inhibitor acting at the eEF2-mediated translocation step during elongation[64,65]. It has been shown that CHX binds to the exit site (E-site) of the 60S ribosomal subunit, in proximity to the binding site of the deacylated transfer RNA (tRNA). The binding of CHX to the E-site inhibits the release of tRNA, and arrests ribosomal translocation. CHX treatment is followed by endonuclease degradation step, where endonucleases degrade free mRNA, while the ribosome-engaged mRNA sequence regions of ~28-32 nucleotides (nt) are protected from degradation. The ribosome protected fragment regions are then sequenced by NGS, and the subsequent mapping of the reads reveals the locations of translating ribosomes on the mRNA[61]. Therefore, the standard CHX protocol of ribosome profiling (Ribo-Seq) can be used for transcriptome-wide assessment of protein translation as it acts on fully assembled translating ribosomes. While this technique reveals accumulation of ribosomes near upstream initiation sites, it is important to note that CHX acts on translating ribosomes, and therefore this procedure reveals all regions in the transcriptome that are undergoing translation[66]. With minor protocol adjustments, this technique can be repurposed to provide information specific to translation initiation[62].

While CHX acts on elongating ribosomes, two protein synthesis inhibitors: Lactimidomycin (LTM) and Harringtonine (HT), have been found to inhibit protein translation, by preferentially acting on initiating ribosomes. LTM, a larger and more potent cycloheximide-related molecule isolated from *Streptomyces* similarly to CHX binds the E-site. However, due to its larger size, LTM is restricted to binding to the E-site only when it is empty[67,68]. Given that this occurs only during initiation, when the Met-tRNA$_i^{Met}$ enters the P-

site, LTM preferentially binds to initiating ribosomes and cannot bind to elongating ribosomes with tRNA-engaged E-sites[62]. The natural *Cephalotaxus* alkaloid HT is a potent protein synthesis inhibitor that preferentially targets initiating ribosomes through a mechanism that involves the binding of free 60S subunits, and a subsequent locking of the ribosome in place during 80S complex assembly[62]. Transcript initiation sites (TIS) can be enriched by treatment with puromycin, which facilitates the release of peptide-elongating ribosomes[69].

Ribosome profiling studies have revealed an astonishing complexity of translation initiation in the 5′ UTR, challenging the long-time belief that translation typically initiates at well-defined AUG initiation codons. A significant portion of the proteome (~60%) has been found to initiate at non-AUG initiation codons[70]. While strong evolutionary pressure enriches the levels of AUG-initiated proteoforms across all organisms, NCCs are more prevalent in the 5′ UTR[71]. The decreased evolutionary pressure on some NCCs is evident from their enriched presence in the 5′ UTR in comparison to the AUG codon (Discussed in Results 1). If codon efficiency is considered, it can be expected that non-AUG proteoforms are present at lower concentrations in comparison to their associated pORF[71]. However, the decreased initiation efficiency at non-AUG codons doesn't discount the functional potential of the proteins they initiate, their regulatory potential in context of expression of the pORF, or more generally, their initiating potential under different cellular conditions.

While the prevalence of non-AUG initiated proteoforms is a point of scientific discussion, several functional non-AUG initiated proteins have been described in mammals. A notable example of non-AUG initiation is the oncogenic transcription factor *MYC* (c-Myc), which has been long known to initiate at a CUG NCC[72]. Two proteoforms are translated from the c-Myc mRNA – a shorter protein initiating at an AUG initiation codon, and a longer (p68, myc1) N-terminally extended protein which initiates at a near-cognate CUG initiation codon, 15 codons upstream of the AUG initiation codon. Several additional genes with CUG-initiated

proteoforms including the secreted vascular endothelial growth factor (*VEGFA*), Bcl-1-associated anthanogene-1 (*BAG1*), fibroblast growth factor (*FGF*), and the phosphatase and tensin homolog (*PTEN),* have been implicated in cancer[58]. While the initiation in these examples introduces similar proteoforms with variable N-termini if the initiation codon is not in the same frame as the pORF it can generate a protein with no sequence homology to the main proteoforms. It was recently discovered that an out-of-frame CUG initiation in the polymerase gamma (*POLG)* gene results in the pORF overlapping translation of the *POLG* alternative reading frame (POLGARF) – a 260 amino acid protein implicated in extracellular signaling[73]. Interestingly, all these experimentally validated examples use CUG as an initiation codon. It has been shown that a subset of non-AUG codons can initiate translation as efficiently as AUG codons, with CUG being the most efficient, followed by ACG and GUG[71,74].

Non-AUG initiation has been implicated in repeat expansion-related neurological disorders through the RAN (repeat-associated non-AUG) translation – an unconventional initiation mechanism resulting in the production of toxic dipeptide repeat proteins (DPRs)[75]. RAN translation has been associated with notable neurological disorders including Frontotemporal dementia (FTD) and amyotrophic lateral sclerosis (ALS), but may play a broader role in central nervous system (CNS) disorders such as Huntington's disease (HD), spinocerebellar ataxia type 8 (SCA8), myotonic dystrophy types 1 and 2 (DM1 and DM2) and fragile X tremor ataxia syndrome (FXTAS)[76-80]. RAN translation is promoted by GC-rich repeat expansion loop structures and can initiate at non-AUG codons before or internal to the structures resulting in the translation and accumulation of neurotoxic DPRs. The non-specific initiation results in a diverse pool of DPRs, which is further diversified by the ability to generate DPRs from the antisense transcript[80].

While the specific mechanism for each of these disorders has not been completely elucidated, ALS/FTD C9orf72 C9RAN initiation has been found to be acting through a cap,

eIF4E, and eIF4A-dependent mechanism. Depending on the reading frame, RAN translation in C9orf72 has been observed to start either at a NCC CUG upstream of the GC-rich structures (G_4C_2 or CGG repeats) or within the repeat structure itself. It has been reported that the replacement of the m^7G cap with an A-cap analog incapable of recruiting eIF4E, results in decrease of C9RAN translation in all the reading frames. Similarly, treatment with hippuristanol, a selective inhibitor of eIF4A, has been shown to result in reduction of C9RAN translation[81].

In summary, non-AUG initiation is prevalent in the transcriptome and can occur through a variety of mechanisms. Aberrant non-AUG initiation has been associated with cancer and neurodegeneration, suggesting that understanding the functional role and mechanisms of non-AUG initiation may be of therapeutic interest. Numerous viral proteins with critical functions are derived from non-AUG initiation codons[82]. Viruses have evolved to optimize their translatability in post-infection stress conditions. Some of these elements, such as internal ribosomal entry sites (IRES), enable optimized translation of viral proteins under conditions where translation of cellular mRNAs is inhibited, and can promote non-AUG initiation[83]. The translation of the second ORF of the Cricket paralysis virus (CrPV) RNA, one of the best understood mechanisms of viral initiation at non-AUG codons, highlights the ability of viruses to initiate at non-AUG codons without the use of any initiation factors[84-86]. This raises interesting questions with regards to cellular translation under post-infection conditions, as non-AUG initiation may be an important translational avenue for the cellular stress response.

1.1.1.4 Noncanonical translation initiation

While most of eukaryotic translation initiation can be attributed to the cap-dependent leaky scanning model, this mechanism is often inhibited during viral infection and cellular stress[35,87-89]. Namely, multiple viruses encode proteases that target eIFs; some specifically

target the scaffold eIF4G[90,91]. When unable to efficiently translate their proteome through the cap-dependent scanning mechanism, cells must rely on alternative mechanisms, and utilize internal ribosome entry sites (IRES) to respond to cellular stresses[88,89].

IRES sequences are structural elements first discovered in viruses, capable of recruiting ribosomes to the internal region of an mRNA[92,93]. Due to their conceptual challenge to the leaky scanning model, their diverse sequence composition, and the imperfect methods used to measure their activity, their function was initially disputed[35]. However, it is now well-established that IRESs are a common mechanism for translating viral proteins and are also present in the sequence leaders of cellular mRNAs involved in stress responses including: apoptosis, hypoxia, and nutrient depletion[94]. Increasing evidence for the translation of cap-lacking circular RNAs provide more confidence in this cap-independent mechanism of cellular IRES[88,95]. Based on their reliance on IRES-transacting factors (ITAFs), initiation factors (IFs) and their RNA secondary structure, viral IRESs are classified into four categories. The first category of IRESs can directly recruit 40S ribosomes without relying on an initiation codon, ITAFs or IFs, or the use of Met-tRNA$_i^{Met}$, as they are capable of mimicking the structure of tRNA$_i$[88,89]. As their structural complexity increases, IRESs increasingly rely on IFs and ITAFs to recruit 40S ribosomal subunits (**Table 1**)[88,89].

	Group I	Group II	Group III	Group IV
IFs	None	eIF2 eIF3	eIF2 eIF3 eIF4A eIF4B eIF4G	eIF2 eIF3 eIF4A eIF4G
ITAFs	None	None	PTB ITAF$_{45}$	PTB PCBP1 PCBP2
40S Recruitment	Direct binding	Direct binding	ITAF-assisted binding	ITAF-assisted binding
40S Scanning	Scans	Does not scan	Does not scan	Scans

Table 1. Classification of viral IRESs

Viral IRESs are classified based on their dependence on IFs and ITAFs. Groups I and II are characterized by the inclusion of three pseudoknots (PKI, PKII, and PKIII) and multiple stem-loops which can mimic an initiation codon, allowing for a direct interaction with the 40S subunit without relying on ITAFs and IFs. Table adopted from Yang *et al.*[96]

In contrast to viral IRESs, eukaryotic IRESs belong to one of two categories: i) type I, which relies on *cis*-recruited ITAFs, and ii) type II which contain 18S-binding *cis*-elements. Approximately 10% of the human transcriptome is predicted to contain IRES sequences[97].

A second cap-dependent mechanism of translation initiation, which diverges from the scanning model is ribosome shunting, a process by which ribosomes can completely bypass regions of the 5′ UTR to reach an initiation codon. Ribosome shunting may provide insight into why some mRNAs exhibit efficient translation, despite the presence of complex secondary structures, or *cis*-regulatory elements in their 5′ UTR[98]. This mechanism is proposed to depend on the interaction of shunt donor and acceptor sequences that promote the skipping of a flanked inhibitory element without scanning through it[99-101]. While ribosome shunting has been observed to be active under stress conditions in only a handful of cellular eukaryotic mRNAs (*BACE1*, *cIAP2*, *HSP70*), it is likely that future work will reveal more shunting sites in the transcriptome[102].

An increased number of IRES elements have been associated with selective protein synthesis during cellular stress conditions[103,104]. A recent computational study utilizing human datasets has annotated 17,013 predicted IRES elements, suggesting that IRESs may be represented in more than one third of the transcriptome[103]. Understanding the mechanisms of noncanonical translation can be critical for unravelling the nature of the cellular stress response, and may be harnessed for therapeutic development.

1.1.2 Eukaryotic Translation: Elongation

Elongation, the second phase in protein translation, can be divided into three steps: decoding, peptide bond formation, and tRNA-mRNA translocation. Unlike the initiation and termination phases, the translation elongation phase is highly conserved and similar across all kingdoms of life[105]. Consequently, the two elongation factors that play a critical role in eukaryotic elongation: eukaryotic elongation factor 1 A (eEF1A) and the GTPase eukaryotic elongation factor 2 (eEF2), are orthologs of the prokaryotic elongation factor thermo unstable (EF-Tu) and elongation factor G (EF-G) – also known as translocase[6].

The process of initiation ends with the assembly of the 80S ribosome, empty A and E-sites, and an initiating tRNA engaged at the P-site paired with an initiating mRNA codon found in an optimal initiation context (**Figure 2c**). In the next step, an aminoacyl-tRNA(aa-tRNA) must position itself at its corresponding codon at the A-site. The aa-tRNA is presented at the A-site by eEF1A in the form of a ternary complex eEF1A•GTP•aa-tRNA, by interacting with the factor binding domain on the 60S ribosomal subunit (**Figure 3a**). The codon-anticodon pairing, which ensures that the proper tRNA is selected, is regulated by the 18S rRNA of the small ribosomal subunit[106]. Conformational constraints of the 18S rRNA guide the proper selection of the cognate aa-tRNA from a pool of cognate and non-cognate ternary complexes that compete for binding the A-site[106]. Correct codon-anticodon pairing promotes the

interaction of the small ribosomal subunit rRNA with the mRNA-tRNA duplex, which in turn initiates the hydrolysis of the eEF1A•GTP•aa-tRNA ternary complex-bound GTP. This causes a structural rotation which releases eEF1A•GDP from the ribosome[105]. This process brings the amino acid at the A-site into close proximity with the incoming amino acid at the P-site, while the E-site remains empty[6].

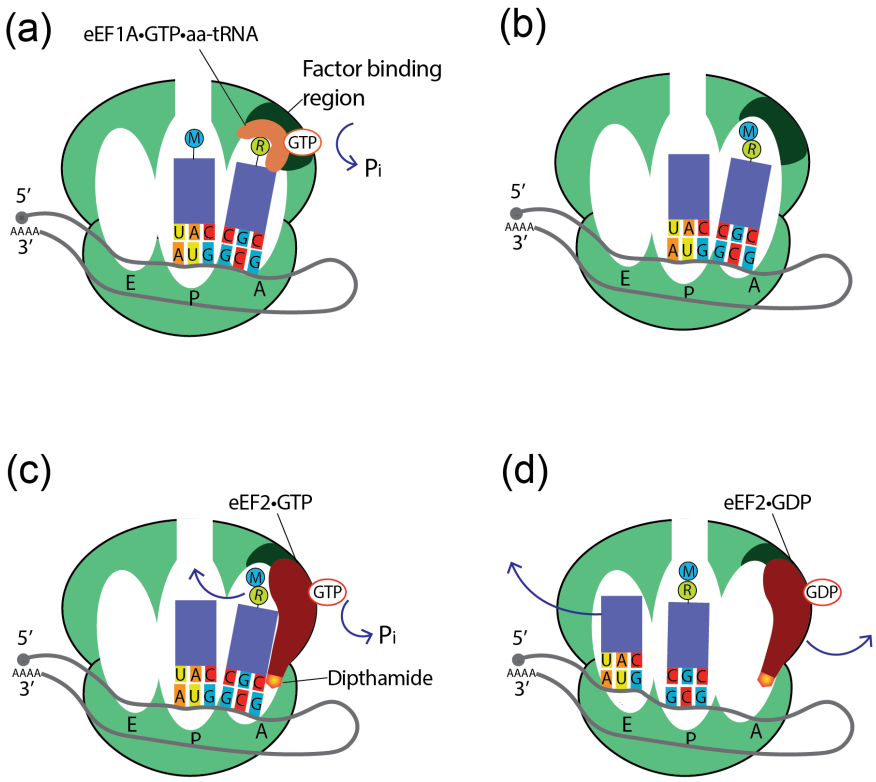


Figure 3. Translation elongation

A diagram representing the process of elongation. (a) The assembled 80S ribosome is shown at a state where the aa-tRNA is engaged at the A site as a eEF1A•GTP•aa-tRNA ternary complex. eEF1A•GTP interacts with the factor binding region (dark green). (b). The release of eEF1A•GDP promotes the peptide bond formation between the amino acid at the P-site and that of the incoming aa-tRNA at the A-site (c) The translocation of the tRNAs is mediated by the hydrolysis of eEF2•GTP (d) The dissociation of eEF2•GDP allows the 40S subunit to rotate back, which releases the deacylated tRNA from the E-site.

The next step in the process of elongation is peptide bond formation catalyzed by the 28S rRNA of the 60S ribosomal subunit. Following peptide bond formation, the amino acid of the P-site tRNA (initially Methionine) is transferred to the A-site tRNA, where the peptide bond initiates amino acid chain formation by linking the two amino acids (**Figure 3b**). As a result of the deacylation, both the deacylated tRNA and the peptidyl-tRNA are in a hybrid positional state[6]. The acceptor end of the deacylated tRNA is in the E-site of the large ribosomal subunit, while its anticodon is in the P-site of the small ribosomal subunit. Similarly, the peptidyl-tRNA has its acceptor end in the P-site of the large ribosomal subunit, and its anticodon in the A-site of the small subunit[107]. These hybrid states are resolved during the translocation step which is mediated by the second critical elongation factor eEF2•GTP[6].

During translocation the eEF2•GTP binds to the factor binding region of the large ribosomal subunit, bring its dipthamide modification in proximity to the A-site, and disrupts the 18S rRNA interaction with the tRNA and mRNA at the A-site (**Figure 3c**). When this rRNA/mRNA/tRNA interaction is disrupted, the tRNA at the A-site moves to the P-site, which causes the 40S subunit to rotate into a conformation that allows the amino acid chain to enter the protein exit channel, and the deacylated tRNA to move to the E-site. The energy for this translocation is provided by the hydrolysis of the eEF2-bound GTP to GDP[108,109]. The subsequent dissociation of eEF2•GDP causes the 40S ribosome to rotate back and release the deacylated tRNA from the E-site (**Figure 3d**). This returns the ribosome to a state where the E-site is empty, the P-site is engaged with the peptidyl tRNA with the amino acid chain engaged in the protein exit channel, and the A-site is empty and ready to accept the incoming eEF1A•GTP•aa-tRNA. This process is repeated until a termination codon is reached at the A-site.

1.1.2.1 Rate-limiting features of translation elongation

The primary rate-limiting step of protein translation is initiation. Studies focusing on the rate-limiting steps of protein translation have shown that the initiation step is critical in determining the density of ribosomes on an mRNA[51]. In addition to this, the overall rate of protein translation is also regulated at the elongation step. The major determinant for the rate of translation elongation is the availability of cognate tRNA[110]. It has been proposed that this rate is influenced by the competitive nature in the recruitment of a tRNA at the A-site[111]. More specifically, some studies have pointed out that the rate of elongation is influenced by the competition of cognate and near-cognate tRNA interactions[112]. Recent studies have shown that the amino acid composition of the synthesized protein can significantly impact the rate of translation elongation[113]. This observation is further supported by studies focusing on eEF5A, which has been found to assist with the efficiency of peptide bond formation by binding to the E-site when the ribosome encounters problematic motifs and stalls[114]. More recently, a mouse study surveying elongation *in vivo* revealed that elongation rates can differ by more than 50% across different organs, and further showed that elongation rates decrease with aging. This demonstrates that protein levels are also regulated during the elongation step of protein translation[115].

1.1.3 Eukaryotic Translation: Termination

Translation termination is the process of releasing a completed amino acid sequence and marks the completion of elongation. The process of termination is initiated by the encountering of one of three stop codons (UAA, UAG, UGA) at the A-site, which forms the pre-termination complex. In eukaryotes, the termination process requires two release factors (RFs), eRF1 and the GTPase eRF3, which bind to the A-site and ribosome factor binding region as an eRF1•eRF3•GTP complex (**Figure 4**)[6]. In comparison with prokaryotic stop codon

recognition that requires multiple RFs, eRF1 can recognize all three stop codons[116]. When a stop codon is encountered at the A-site the ternary complex eRF1•eRF3•GTP is positioned in such a way that the N-domain of eRF1 is in contact with the stop codon at the A-site, but its M-domain, responsible for cleaving the polypeptide chain is not close enough to the cleavage site of the tRNA at the P-site (**Figure 4a**). This structural conformation causes the sarcin-ricin loop (SRL) region at the factor binding domain of the 60S subunit to activate the GTPase activity of eRF3, which hydrolyses the GTP of the eRF1•eRF3•GTP ternary complex to GDP and irreversibly stabilizes the eRF1 interaction with the stop codon at the A-site (**Figure 4a**)[6,117]. In addition to locking eRF1 at the A-site, GTP hydrolysis moves the M-domain in proximity to P-site tRNA, which increases the accessibility of the polypeptide ester bond to a nucleophilic attack. This results in the release of the polypeptide chain, and the dissociation of the eRF3•GDP complex (**Figure 4b**)[6,117]. The ribosome at this stage forms the post-termination complex where eRF1 is engaged at the stop codon at the A-site, the tRNA at the P-site is deacylated, while the E-site is empty (**Figure 4c**)[6].

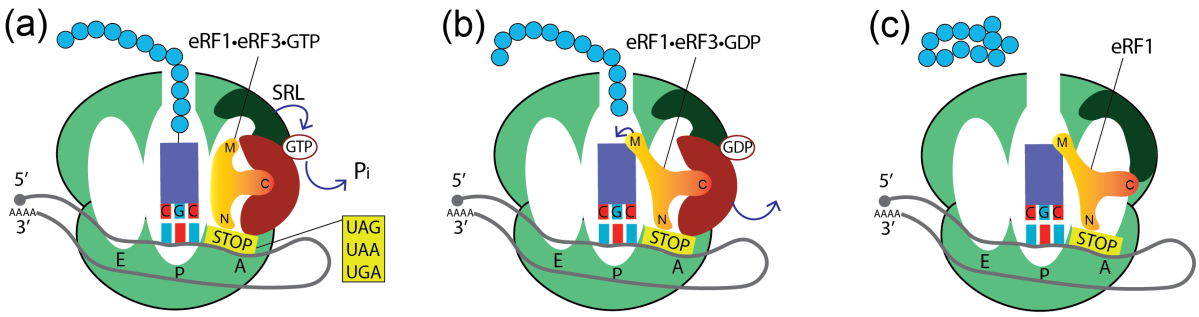


Figure 4. Translation termination

A diagram representing the process of translation termination. (a) When one of the stop codons is encountered (UAG,UAA,UGA), the eRF1•eRF3•GTP ternary complex is engaged at the factor binding region, where the N-domain of eRF1 interacts with the stop codon at the A-site (b) The SRL region of the factor binding domain activates the GTPase activity of eRF3 which hydrolyzes GTP, stabilizes eRF1 at the A-site and brings the eRF1 M-domain in close proximity to the polypeptide chain (c) Following a nucleophilic attack on the polypeptide ester bond the polypeptide chain is released.

1.1.4 Eukaryotic Translation: Ribosome recycling

When protein synthesis is complete, the post-termination complex undergoes a recycling process which involves the disassembly of the 80S ribosome into its constitutive subunits. The process of ribosome recycling is initiated by the recruitment of ABCE1 to the post-termination complex. ABCE1 has a hinge domain which interacts with the C-domain of the eRF1 at the ribosome subunits junction[118-120]. This interaction happens when ABCE1 is in an open conformation, where its two pockets, each with the capacity to bind an ATP molecule, are empty[118-120]. When two ATP molecules bind to the ABCE1 complex, the complex conformation changes to closed, the phosphate bond of the ATP is cleaved, and the released energy acts through the hinge domain to release the 60S subunit and eRF1[118].

The subsequent release of the tRNA and mRNA from the 40S subunit relies on initiation factors that act through two pathways. The first mechanism requires the binding of eIF1 and eIF1A to the empty E-site and A-site respectively, and the additional assistance from eIF3 to dissociate the tRNA[118,121]. This mechanism relies on the ability of eIF1 to discriminate against non-initiating codons and destabilize the non-initiator tRNA at the P-site[118,122]. The second mechanism depends on eIF2D (Ligatin) to destabilize the codon-anticodon pairing at the P-site, which results in the release of the tRNA and mRNA[121]. Two additional proteins MCT1 and DENR, have structural homology with the amino and carboxy-termini of eIF2D, and are known to be able to destabilize the P-site with lower efficiency[123].

Ribosome profiling evidence of translation of pORFs in genes with complex 5′ UTRs, containing multiple initiation sites, suggests that a proportion of the 40S subunits remain viable for further scanning and initiation[124]. This means that factors required for the complete disengagement of the 40S subunit from mRNA may compete with Met-tRNA$_i^{Met}$•eIF2•GTP TCs and factors which would make the 40S subunit competent for scanning and reinitiation. Under such a model, a proportion of the 40S subunits may stay engaged, while others may

dissociate from the mRNA. The continuation of scanning by 40S subunits may be supported by the circularized conformation of mRNAs during the process of translation, which is facilitated by the interaction of the eIF4G-bound PABP with the poly-A tail. The binding of PABP to both eIF4G at the cap structure and the poly-A tail, may serve as a mechanism to loop 40S subunits back in proximity to the 5′ cap.

1.2 Post-transcriptional regulation and the correlation between the human transcriptome and the proteome

The central dogma defines the flow of information from DNA to messenger RNA (mRNA) to protein[125]. The implicit underlying assumption has been that differences in protein levels can be explained by mRNA levels. However, over the past two decades, multiple studies focused on describing the relationship between the transcriptome and the proteome, have reported diverse results[126]. Early studies have reported a broad range of correlation, from weakly correlated to strongly correlated; with the study with the largest dataset (N=4251) showing a moderate correlation: Spearman rank coefficient $r_s = 0.57$[127,128]. These studies relied on technologies such as microarray and quantitative SDS-PAGE/Western blots, which have been phased out in favor of highly accurate next generation sequencing (NGS) and liquid chromatography coupled with mass spectrometry (LC-MS)[129]. However, while these improvements have increased the resolution and accuracy of more recent datasets, the reported correlations still fall in a broad range between 0.7-0.52[126,130-133]. These results suggest that while transcript levels are a key contributing factor in determining the variance in protein levels, the cellular abundance of proteins is significantly determined by additional post-transcriptional regulatory elements. Based on their mechanism of action, these elements can be classified into *cis*-acting or *trans*-acting.

1.2.1 Post-transcriptional regulation by *trans*-acting elements

Protein translation is highly regulated by RNA binding proteins (RBPs). Conceptually this is not surprising as multiple RBPs (notably: eIF4E, PABP) are directly involved in the process of translation. The RBPs participating in the translation are a small subset of a much larger group of RNA-interacting molecules. The human genome encodes more than 1,500 RBPs that target coding and non-coding regions. RBP binding sites have been shown to have increased conservation, which is indicative of selection for RBP regulation[134]. RBPs are a class of *trans*-acting regulatory proteins that can bind to specifically recognized sites in the mRNA. Some of these motifs have been found across mRNAs of multiple genes, suggesting that certain RBPs can regulate the expression of multiple genes. The effects of RBP on the mRNA can be diverse. It has been shown that RBPs can function during mRNA maturation, affect cellular distribution, increase or decrease mRNA stability, and increase or decrease protein translation[135,136].

Another class of trans-acting elements that can regulate gene expression are microRNAs (miRNAs). miRNAs are endogenous, single-stranded noncoding RNAs capable of regulating gene expression through their association with micro-ribonucleoprotein (miRNP) complexes, and Watson-Crick interaction with cognate regions that are typically found in the 3′ UTR of target mRNAs[137]. miRNAs were jointly discovered by the Ambros and Ruvkun groups[138,139]. The Ambros group observed that *lin-4*, a gene involved in regulating the larval stages in the nematode *C.elegans*, does not code for a protein, but instead transcribes a pair of short RNAs (22nt and 61nt in length respectively)[139,140]. Concurrently, the Ruvkun group showed that *lin-4* regulates Lin-14 expression by interacting with complementary regions in the *lin-14* 3′ UTR[138]. It is now widely recognized that *lin-4* was the first of a diverse and prevalent class of gene expression regulators.

The biogenesis of miRNAs starts with longer stem-loop pri-miRNAs transcripts, transcribed by RNA polymerase II (Pol II) and are initially processed by the Drosha (DROSHA) enzyme complex to precursor miRNAs (pre-miRNAs). Pre-miRNAs are then exported to the cytoplasm where they are processed by the RNA-induced silencing complex (RISC), which is composed of three proteins: Argonaute 2 (Ago2), Dicer, and the transactivation response element RNA-binding protein (TRBP)[141,142]. This processing involves the cleavage of pre-miRNA and selective loading of one of the miRNA strands (guide strand) into Ago2[141,143]. The loaded miRNA binds to its partially complementary target and either inhibits translation, or promotes mRNA degradation in processing (P)-bodies[144].

miRNAs can regulate mRNAs through both cognate and near-cognate sequence interactions. Full sequence complementarity with the site of interaction results in mRNA cleavage and subsequent degradation, while partial complementarity promotes decapping and deadenylation of the mRNA; both resulting in downregulation of gene expression. The ability to exert their effect with partial complementarity enables miRNAs to regulate groups of transcripts[137,145].

1.2.2 Post-transcriptional regulation by *cis*-acting elements

Cis-acting elements found in the mRNA transcripts of all genes have been found to significantly influence the variance of translation rates across organisms. General translational *cis*-elements have been found to regulate the recruitment, scanning, and assembly of the ribosome, as well as translation[126,146,147]. This group of regulatory elements includes the Kozak consensus sequence, mRNA folding, protein coding sequence (CDS) length, codon usage, and upstream open reading frames (uORFs)[148]. While the individual quantitative contribution of these elements to the variance in protein levels has not been well characterized, altogether *cis*-elements are estimated to account for 42-81% of the variance in yeast, plants, and mammals[148].

The critical aspect of the Kozak consensus sequence in context of initiation was previously discussed. Initiation codons with weak Kozak sequence context have a lower probability of recruiting scanning ribosomes. Thus, strong Kozak sequences promote higher translation rates, while weak Kozak sequences result in unproductive scanning ribosomes which bypass the pORF initiation codon; as described by the leaky scanning model[35].

Local regions of mRNA sequences can interact through Watson-Crick pairing to create secondary hairpin structures. Consistent with the scanning model, both scanning and translating ribosomes must unwind these structures. Hairpins or higher-order secondary structures have been known to inhibit initiation, especially if they occur in proximity to the cap structure[149,150]. However, the presence of RNA regions adopting stem-loop secondary structures has also been implicated in ribosomal pausing, frame shifting, termination code readthrough, RBP recruitment, and regulation of mRNA half-life[151,152]. Notably, secondary structures with a median length of ~174 nucleotides can provide alternative translation initiation avenues by defining structural regulatory motifs such as IRESs[153]. Shorter stem-loop structures containing the CAGUGN iron-responsive element (IRE) motif found in the 5′ or 3′ UTRs have been found to regulate translation of multiple proteins including: ferritin (FTH1), ferroportin (Fpn), transferrin receptor (TfR), divalent metal transporter 1 (DMT1), amyloid precursor protein (APP) and α-synuclein (α-Syn), by recruiting the iron regulatory protein (IRP)[154]. In addition to these functions, a stem-loop structure in the 3′ UTR of histone mRNA recruits the stem-loop binding protein (SLBP) which can serve as a substitute for a poly(A) tail, and is also involved in the rapid degradation of histone mRNA[152]. Similarly, AU-rich elements (AREs) found in the 3′ UTR, and typically containing multiple copies of the AUUA motif, can accelerate mRNA decay[155].

Length of the pORF CDS has been inversely correlated with translation rates, as 60S ribosomal subunits get recycled at the CDS termination codon, while the 40S ribosomal subunit

must likely scan to the poly-A tail region to return in close proximity to the 5′ cap structure[148,156]. Therefore, mRNAs with shorter CDSs, where the termination codon is closer to the pORF initiation codon, have faster ribosome recycling rates, which results in higher translation rates[156].

Most amino acids are encoded by several synonymous codons which are not represented in the pool of tRNAs with equal frequencies. This imbalance in synonymous codons in the pool of aminoacylated tRNAs is likely the result of evolutionary selection acting to optimize translation[157]. The preferential usage of the more prevalent "optimal" codons is known as codon usage bias[158]. Codon usage bias is a characteristic of prokaryotic and eukaryotic organisms, and has been proposed to influence translation efficiency, the accuracy of the translated peptide, and protein folding[157]. The underlying hypothesis regarding codon usage is that codon identity influences the rate of elongation. Highly expressed genes are typically encoded by codons which correspond to abundant tRNAs[157]. This concept has been utilized as a strategy to optimize protein translation in overexpression systems for many years, and was recently applied to the mRNA Covid-19 vaccines[159].

1.3 Upstream Open Reading Frames (uORFs)

More than 50% of all human transcripts contain one or more canonical (AUG) initiation codons in their 5′ UTR sequences[160]. When these initiation codons are paired with an in-frame stop codon they create ORFs, which due to their sequence position relative to the pORF are referred to as upstream ORFs (uORFs) (**Figure 5a**). While typically found in the 5′ UTR, uORF termination codons may also be located in the CDS or 3′ UTR (**Figure 5b**). The presence of uORFs in the 5′ UTR has been associated with lowered pORF protein levels, suggesting that uORFs are *cis*-regulatory elements that have an inhibitory effect on translation of the pORF[161,162].

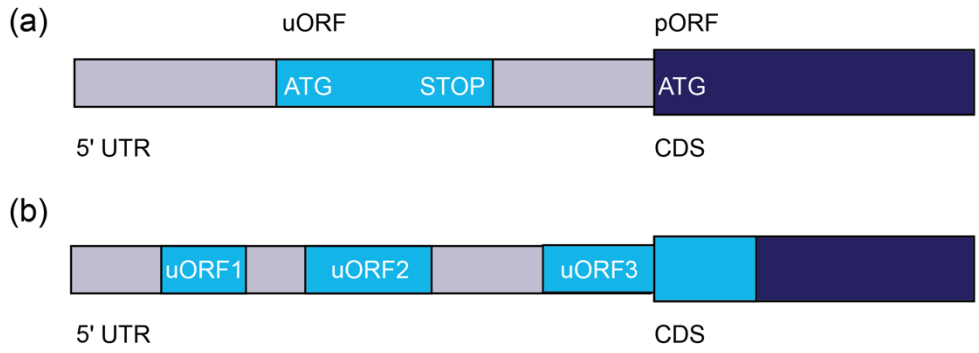


Figure 5. Upstream Open Reading Frames (uORFs).

A schematic representation of the 5′ UTR and beginning of the coding sequence (CDS) of the primary open reading frame (pORF). (a) An example of an uORF in the 5′ UTR. (b) Example of multiple uORFs in the 5′ UTR, with uORF 3 terminating in the CDS of the pORF.

The inhibitory function of uORFs was first observed by Kozak, who later demonstrated that a certain intercistronic distance is required for scanning ribosomes to regain reinitiation competence after translating a uORF[163-165]. The influence of this positional feature, as well as the global inhibitory effects of uORFs were later confirmed in bioinformatics studies analyzing mouse proteomics datasets[162,166]. Namely, specific uORF features (uORF context, conservation) and 5′ UTR architectures (number of uORFs, uORF position) have been associated with a decreased expression at the pORF[162,166-168]. In addition to their inhibitory function, some 5′ UTR uORF architectures have been found to promote translation of the pORF under cellular stress conditions[169-174]. Lastly, some uORFs have been found to be inconsequential to the translation of the pORF[175,176]. Considering their broad regulatory impact on translation, it can be expected that mutations that deregulate uORFs may play a critical role in disease[1,177]. Therefore, understanding the mechanisms of uORF translational regulation is of significant therapeutic interest, as they represent potential targeting regions for modulating gene expression, in particular protein upregulation.

The exact definition of uORFs is not yet settled, as some groups have set specific feature constraints based on their own observations. For instance, the most comprehensive bioinformatics study of uORFs defines them as sequence features with a minimal length of 9 nt, where the first codon is an upstream AUG (uAUG) which is also out-of-frame with the pORF, while the last codon is an in-frame stop codon that ends before the CDS[162]. In addition, ribosome profiling studies have revealed that a significant portion of the proteome initiates upstream of the pORF, at both AUG or non-AUG initiation codons, suggesting that definitions based on the canonical AUG codon may not capture all uORFs [62,70,178]. For the purposes of the bioinformatics analysis presented herein, uORFs are defined as *cis* sequence elements present in the 5′ UTR of the mRNA that are formed by a canonical AUG initiation codon associated with an in-frame termination codon. Separately, the preferential usage of alternative near-cognate initiation codon is considered, recognizing that uORFs may initiate at non-AUG sites[179].

Bioinformatics analyses reveal that uORF initiation codons appear at a lower frequency than expected in the 5′ UTR and are highly conserved[162,180,181]. However, most uORF encoded peptides are not conserved, which suggests that uORF primarily act through a *cis* mechanism[180]. Thus, it has been proposed that uORF initiation codon conservation has occurred to solely preserve their translation inhibitory function[180]. Interestingly, some uORF configurations which are repressive under normal conditions, have been found to promote pORF translation under stress conditions[169,182]. This suggests that uORFs may also serve as translational switches that can trigger the translation of genes that activate the stress response.

1.3.1 Ribosome uORF interactions

The mechanism by which uORFs inhibit translation is incompletely understood. However, several options have been proposed with regards to the mechanisms by which uORFs

regulate translation: (i) the ribosome can assemble at the uORF initiation codon and translate the uORF peptide (**Figure 6a**), (ii) the ribosome can initiate at the uORF and undergo complete ribosome recycling , (iii) the scanning ribosome can scan through the uORF sequence and initiate at the pORF according to the leaky scanning model (**Figure 6c**), (iv) The ribosome can translate the uORF, recharge with factors to become competent for subsequent translation(**Figure 6**Error! Reference source not found.**d**), (v) The ribosome can initiate at the uORF and inhibit translation through *cis* interaction of the translated uORF micropeptide, (**Figure 6e**), or a combination of ribosome collisions and stalling at the uORF may lead to mRNA degradation via the non-sense mediated decay (NMD) pathway (**Figure 6f**)[1].

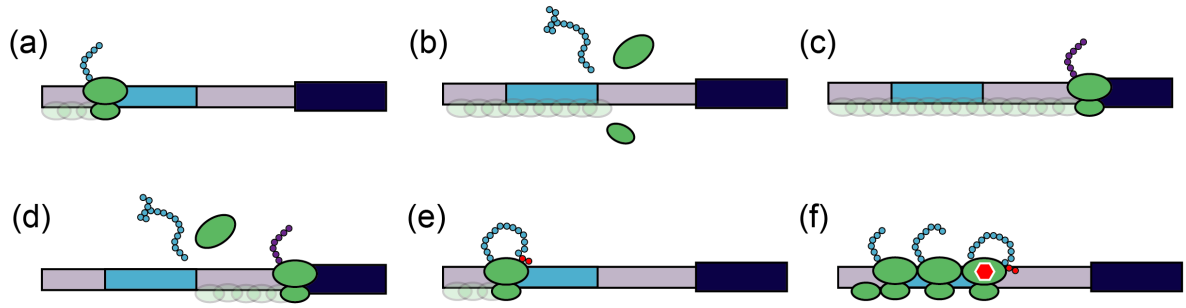


Figure 6. Ribosome uORF interactions and their effect on translation

A schematic representation of ribosome uORF interactions depicting a single uORF (light blue) in the 5′ UTR (grey), and the start of the pORF CDS (dark blue). (a) Initiation and translation at uORF. (b) uORF translation followed by ribosome disassembly (c) Leaky scanning – bypass of the uORF initiation codon. (d) Reinitiation after uORF translation (e) *cis*-inhibition of uORF peptide (f) Ribosome stalling and nonsense mediated decay (NMD)

When translation initiation at an uORF occurs, the inability to reinitiate at the pORF is likely the driving factor of translational inhibition of the pORF. However, initiation at uORFs has also been associated with mRNA degradation through the non-sense mediated decay pathway – a cell surveillance pathway through which mRNAs with premature stop codons are

degraded[183-188]. This suggests that the expression levels of some mRNAs may be regulated by uORFs. Conversely, in some transcripts uORFs may potentially be bypassed by mechanisms other than leaky scanning and reinitiation, including alternative transcription sites driven by secondary promotors, exon splicing in spliced UTRs, or bypass by IRES or ribosomal shunting.

1.3.2 Translation reinitiation after uORF translation

The fate of the ribosome post uORF translation is not completely understood, but several possible mechanisms have been discussed[1]. As with standard pORF translation, the ribosome can completely dissociate **(Figure 6b**). The outcome of either partial or complete ribosome dissociation results in a decrease in the number of translation-competent ribosomes after the uORF termination codon. Depending on initiation context, the ribosomes may scan through uORF initiation sites (**Figure 6c**). This hypothesis, originally proposed by Kozak, is consistent with the leaky scanning theory, where with some probability the ribosome may bypass an initiation codon if the initiation context is not optimal[189]. Under these conditions, which are guided by the strength of the initiation context, and the competence of the scanning ribosome at the time it encounters an uORF initiation codon, ribosomes initiating at uORFs become non-productive, and any residual levels of pORF translation can be explained by the ribosomes which have scanned through the uORF initiation codon.

The level to which complete ribosome disassembly occurs at the uORF termination codon is not known. While this likely happens to some extent, if the complexity and high energy cost of generating initiation-competent ribosomes is considered, this would be an inefficient utilization of ribosomes for the cell. This raises questions regarding the purpose of translation of uORFs and the possible functional role of some of the translated uORF peptides[190]. However, weighed against overproducing much larger peptides of the pORFs, the cost of translating shorter uORF peptides in the 5′ UTR is smaller.

The effect of uORF to pORF distance on protein levels suggests that ribosomes translating uORFs can reinitiate at downstream sites which may be the pORF, or another uORF in the 5′ UTR. This mechanism would imply that not all 40S subunits dissociate from the mRNA during the ribosome recycling step, and that 40S subunits can with some probability get recharged with the necessary factors to reinitiate at a subsequent initiation site (**Figure 6d**)[121,161]. It has been proposed that downstream re-initiation is dependent on *trans*-acting factors, mRNA structure, sufficient distance between the uORF stop codon and the pORF; with longer sequence distance likely providing the necessary time to recharge the scanning ribosome with a ternary complex and initiation factors[161]. While the mechanism of reestablishing initiation competency of the 40S subunit is not fully understood three general pathways have been proposed that involve canonical factors, reinitiation factors, or 80S ribosomes that don't separate during ribosome recycling [121].

1.3.2.1 Canonical initiation factor-mediated reinitiation

Early hypotheses for translation reinitiation predicted that some initiation factors may dissociate stochastically, and as such may remain associated during the elongation step[163]. Ribosomes that retain these critical initiation factors were hypothesized to be competent for reinitiation. While this has not been conclusively proven, the ability of eIF3 to bind to the solvent surface of the 40S subunit, may facilitate the retention of eIF4F[191]. A recent study in yeast, revealed that eIF4G and eIF4A, but not eIF4E are retained on ribosomes terminating at uORFs, suggesting that only some, but not all members of the eIF4F complex are required for reinitiation[192]. This observation is consistent with the affinity of eIF4E to bind to the cap structure and suggests that reinitiation is not a cap-dependent process, while recognizing that the initial recruitment of the 40S subunit has likely occurred at the cap structure[54]. The most likely key driver of reinitiation is the acquisition of a Met-tRNA$_i^{Met}$•eIF2•GTP ternary

complex, as it is required for the recognition of an initiation codon. In one of the earliest reinitiation studies, Kozak observed that reinitiation is not limited to the closest AUG codon, and that an intercistronic distance between the uORF stop codon and the subsequent AUG codon was required for successful reinitiation. She estimated that to completely cancel the repressive effects of the upstream uORF its stop codon had to be more than 79 nucleotides away from the pORF[163]. This observation has implications in understanding which uORFs are active in more complex 5′ UTRs which contain multiple uORFs. However, it should be noted that if the reinitiation depends on the availability of the concentrations ternary complex and additional required initiation factors, the exact distance may be variable across cells lines and tissues. Early studies have also implicated uORF length as a feature critical for reinitiation, with the translation of longer uORFs being less likely to result in reinitiation-competent 40S subunits[193,194]. It has been proposed that a longer elongation step provides more time for the dissociation of eIF3 and eIF4F-complex eIFs[121]. Interestingly uORF length was not found to be a statistically significant feature for pORF repression in a computational analysis on the mouse proteome[162].

The canonical factor-dependent reinitiation model is supported by the observation that the minimum required elements for reinitiation are: eIF1, eIF1A, eIF2, eIF3 and Met-tRNA$_i^{Met}$ [195]. In addition to this, it has been shown that the Met-tRNA$_i^{Met}$•eIF2•GTP ternary complex can be reacquired during scanning[172].

1.3.2.2 Noncanonical reinitiation factor-mediated reinitiation

Ligatin (eIF2D), and its C- and N-terminus structural homologs MCT1 and DENR were previously discussed as being capable of destabilizing the 40S-tRNA-mRNA at the P-site. However, in addition to its role in ribosome recycling, Ligatin, MCT1 and DENR have been found to promote initiation downstream of uORF translation in *Drosophila* and human cells[195-

[197]. Several studies have shown that this reinitiation pathway is sensitive to the length of the uORF with conflicting findings in different systems[195-199]. Structural studies suggest that these factors may not deliver the ternary complex, but instead can anchor the tRNA at the P-site, and sterically prevent the binding of canonical initiation factors[196]. More work is required to better understand if the noncanonical reinitiation mechanism is compatible with scanning, and to what extent, if at all, it is dependent on any canonical initiation factors.

1.3.2.3 Reinitiation by post-termination migrating 80S ribosomes

During ribosome recycling the role of ABCE1 is to separate the ribosomal subunits through an ATP-dependent mechanism. In several studies where ABCE1 has been depleted, it has been shown that the ribosomal subunits fail to separate, resulting in the 80S ribosomes that are loosely attached to the mRNA[195]. Upon dissociation of eRF1 from the post-termination complex, the 80S ribosomes can migrate bidirectionally until a codon cognate to the P-site deacylated tRNA is found and are able to reinitiate at that codon[195]. These experiments demonstrate that an incomplete separation of the 80S ribosome may result in a coin-flip like subsequent reinitiation at the nearby upstream or downstream codon cognate to the P-site deacylated tRNA[195,199,200]. Post-termination 80S ribosome migration likely occurs rarely, but structural studies support that ABCE1 is more efficient when both ATP binding sites are in a closed conformation[201]. It may be possible that certain conformations of ABCE1 result in an inefficient separation of the ribosomal subunits. More speculatively, incomplete 80S disassembly may allow for the translation of frame-shifted products, including sequences which may evolve to become novel genes[202].

1.3.3 uORF-dependent translation reinitiation at the pORF

The survival of the cell depends on complex regulatory mechanisms that control the concentrations of biomolecules[1]. While ribostasis and proteostasis are required to maintain normal cellular function, the cell also requires agile and reversible mechanisms to respond to internal and external stimuli. Recent studies combining datasets from *E. coli*, *S. pombe* and human cells have found that the regulation of these responses is more likely to occur at the transcription level rather than translation due to the much higher stability and abundance of proteins relative to that of mRNA[203]. However, it seems that evolution may have found a way to reversibly influence global translation via uORF reinitiation, without drastic changes to ribostasis.

The regulatory effects of uORFs on translation of the pORF are best exemplified by studies of the *S. cerevisiae* gene *GCN4* - a transcriptional activator of amino acid biosynthetic genes. The 5′ UTR of *GCN4* contains four uORFs, which were found to repress the translation of the pORF. Mueller and Hinnebusch, observed that uORFs closer to the pORF were found to be more repressive, while uORF 1 was intriguingly required for *GCN4* translation under starvation conditions[173].

Subsequent studies have fully elucidated the mechanism by which uORFs regulate the pORF translation of *GCN4* and have shown that ribosomes initiate translation at uORF 1, and can reinitiate at uORFs 2-4, with only a few ribosomes initiating at the pORF; resulting in a decrease of protein levels of the *GCN4* pORF. Under starvation conditions, ribosomes translate uORF 1, but due to the inability to regain translational competency, most scanning 40S subunits bypass the inhibitory uORFs 2-4, resulting in reinitiation at the pORF. Thus, the pORF which is not translated under normal conditions, is translated under stress conditions following reinitiation influenced by translating uORFs[204,205]. Ribosome profiling studies have demonstrated a shift in ribosome footprints from the inhibitory uORFs to the pORF under stress

conditions[61]. This example suggests that depending on uORF context, some uORF-regulated transcripts may be activated at the presence of stress stimuli rather than being repressed, highlighting an important regulatory feature of uORFs.

The explanation for this phenomenon likely lies in the time it takes for the scanning 40S ribosomal subunit to recharge with eIFs and a ternary complex in order to become competent for subsequent initiation. Mammalian cells respond to certain stresses including starvation, viral infections, iron deficiency, and accumulation of unfolded proteins in the endoplasmic reticulum (ER), by phosphorylating the α subunit of eIF2 on serine 51[1,204]. In mammals, depending on the stress stimulus eIF2α can be phosphorylated by at least four kinases: general control non-depressible 2 kinase (GCN2), PKR-like ER kinase (PERK), double-stranded RNA-activated kinase (PKR), and heme-regulated inhibitor kinase (HRI)[22]. The function of eIF2 is to deliver the Met-tRNA$_i^{Met}$ in a Met-tRNA$_i^{Met}$•eIF2•GTP ternary complex (TC) form to the 40S ribosomal subunit in order to form the 43S PIC. As mentioned above, during scanning GTP is hydrolyzed, and eIF2 is released in eIF2•GDP form following recognition of the initiation codon. The subsequent recycling of eIF2, and its ability to recruit a new Met-tRNA$_i^{Met}$, depends on the exchange of GDP with GTP which is catalyzed by the guanine nucleotide exchange factor (GEF) eIF2B[204]. When the α subunit of eIF2 is phosphorylated on serine 51, it prevents eIF2 recycling by inhibiting the recharging of eIF2 with GTP by eIF2B. Moreover, the eIF2(αP)•GDP has a higher affinity towards eIF2B, and by a mechanism of competitive inhibition prevents it from recharging non-phosphorylated eIF2•GDP; resulting in a decrease of available TC[204]. When the availability of TC is sparse, post-translational scanning ribosomal subunits require more time to recruit a TC and may bypass initiation codons until they become competent for reinitiation by recruiting a TC and the required eIFs[1,7,204].

1.3.4 uORFs may promote mRNA degradation

RNA degradation mechanisms play a critical role in the maintenance of cell homeostasis, post-transcriptional regulation of gene expression and keeping the balance between transcription and protein levels. Nonsense-mediated decay (NMD) is a highly conserved mechanism that acts to degrade mRNAs containing premature stop/termination codons (PTC). It has been suggested that this mechanism exists to protect dominant-negative or gain-of-function effects of prematurely truncated proteoforms[188]. The exact trigger for NMD has not been fully characterized, but two potential NMD models have been proposed to function in mammals: the "faux 3′ UTR model" where mRNA degradation is triggered by a longer than expected 3′ untranslated region (3′ UTR) as a consequence of encountering a PTC, and the exon-junction complex "EJC-model" where a prolonged interaction of EJCs ~50-55 nucleotides downstream of the PTC initiates mRNA degradation[188].

If the process of reinitiation post uORF translation is not successful, it can be proposed that the translation of uORFs would result in an unusually long 3′ UTR and lead to NMD. uORF-dependent NMD has recently been observed in plants, demonstrating that this mechanism occurs in a uORF length-dependent manner, with longer uORFs of ~50 amino acids being found to be more likely to initiate NMD[183].

The exact mechanisms of how uORFs destabilize mRNA by NMD have not been completely understood. However, it has been proposed that in the "faux 3′ UTR model", a PTC causes an unusually long 3′ UTR which prevents the required interaction between eRF3 and PABP, and instead promotes an eRF3-UPF1 interaction, which further recruits the UPF2 and UPF3 factors to form a functional NMD complex, and initiate mRNA degradation[183,187,188,206]. Recent evidence from genome-wide transcriptome and exome studies supports the "EJC-model" as the preferred pathway for NMD in mammals[186,207]. At the time of splicing the heterotrimeric EJC protein complex comprised of: eIF3A3, RBM8A and

MAGOH, are deposited 24 nt upstream of the exon-exon junction. Under normal circumstances the ribosome can displace the EJC complex. However, a premature termination codon, more than 50-55 nt upstream of an exon junction, leads to retention of the EJC and the subsequent recruitment of UPF3 and UPF2, which interact with UPF1 at the PTC to initiate NMD[208]. NMD siRNA knockdown experiments evaluating the changes in the transcriptome in HeLa cells have demonstrated that 35-50% of uORF containing transcripts are degraded through the NMD pathway[184,209].

The fidelity of the NMD process has been shown to vary across transcripts, with some evidence that NMD may be remedied by the cell through yet to be characterized "NMD escape" mechanisms[183,186,187,210,211]. Evidence of NMD escape suggest that uORF-containing transcript are not regulated by the NMD mechanism as a class. For instance, it has been shown that the protein translation of the pORF of the cytokine thrombopoietin (TPO) is severely restricted by the presence of multiple uORFs in its mRNA isoforms. However, it has been shown that this lack of translation is not a result of NMD, as *TPO* transcripts can escape NMD[185]. Bioinformatics analysis in this study reveal that uORFs may also be enriched in near-background transcripts which are typically filtered out in RNA-seq pipelines (Discussed in the Results 1 section).

1.3.5 Are uORF-encoded microproteins functional?

Recent ribosome profiling studies have revealed a pervasive translation of uORFs. As a result of these observations, there has been an increased interest in the possibility that microproteins encoded by uORFs may be an unexplored functional group of molecules. One argument in support of this hypothesis is that translation is a resource-intensive process for the cell, and the prevalence of uORFs in the genome is too high for this to be an unproductive avenue. Despite these arguments only a small number of uORF peptides have been

functionally defined. However, it is of note that recent discoveries have revealed that the uORF micropeptides of *MIEF1*, *DDIT3*, *FBX09*, *HMGA2*, and *HAUS6*, form stable physical complexes with the protein translated from their respective pORFs[190]. In addition to this, mutations in the sequence that encodes the uORF peptide in (*DRD3*, *WDR46*, *TGFβ3* and *HT3A*) have been associated with disease phenotypes[1]. Yet, given the prevalence of uORFs in the transcriptome, more evidence is needed before uORF peptides are classified as a functional group.

The best argument against uORF microproteins being functional as a class is their lack of conservation, and a decreased negative selection against mutations in the uORF amino acid sequence in comparison to the initiation and stop codons. The maintenance of uORFs amino acid identity has been found to not be under strong selection[177]. In addition to this, uORF AUG initiation codons in the 5′ UTR of both human and mouse were found to be the most conserved codon across 30 vertebrate species[162]. This suggests that a primary factor for uORF function is translation initiation. Moreover, a significant portion of uORFs encode peptides that may be too short to be functional; some being as short a Methionine followed by a stop codon.

1.3.6 uORF implication in human disease

It has been established that under normal conditions uORFs serve to attenuate translation of the pORF[161,162]. An increased number of uORFs in the 5′ UTR have been associated with decreased protein levels in mouse proteomics datasets[162,166]. It has also been shown that under stress conditions, some uORF configurations in the 5′ UTR may promote translation of the pORF[173,204]. Therefore, variants that create uORFs, remove uORFs, or alter the initiation codon or Kozak motif of existing uORFs, may have consequences on the translation levels of the pORF. The described mechanisms of uORF-controlled translation of

the transcription factors *GCN4* and *ATF4 g*enes suggest that any downstream effect of uORF-related mutations would be highly-dependent on the spatial arrangement of the 5′ UTR[212]. For instance, the distance of the uORF stop codon to the next initiation codon may significantly influence the reinitiation potential of 40S subunits[163]. Interestingly, bioinformatics analyses on large-scale population data have found mutations that create uORF initiation codons or disrupt uORF stop codons to be rare[213,214]. The introduction of a uORF initiation codon is likely to create a uORF and have a repressive effect and may result in pathogenic loss-of-function of the pORF protein. Similarly, the removal of a stop codon may result in acquiring a new stop codon closer the pORF, potentially inhibiting reinitiation at the pORF. If a new stop codon is not found before reaching the pORF, the pORF may be bypassed, leading to unproductive ribosomes[214]. For these reasons, the variants that introduce a new initiation codon or remove a stop codon are subjected to strong negative selection pressure. Interestingly, less than 1% of all mutations entries in the Human Gene Mutation Database (HGMD) are found upstream of the pORF codon, highlighting more generally the negative selection pressure against mutations in the 5′ UTR[215].

Despite these bioinformatics assessments uORF mutations have been associated with specific pathogenic phenotypes. Generally, a disease phenotype may arise due to mutations that either introduce a new uORF, remove a uORF, or modify the uORF peptide encoding sequence. Pathogenic uORF-creating mutations have been described in: *FXII*, *HBB*, *PRKAR1A*, *IRF6*, *SRY*, *SPINK1*, *CDKN2A*, *LDLR*, *CFTR*, *KCNJ11*, *PEX7*, *POMC*, *GCH1*, and *HAMP* genes, while pathogenic uORF-disruptive mutations have been described in the *HR* and *TPO* genes[1].

1.3.7 Therapeutic potential of targeting uORFs

Targeted inactivation of the translation-inhibitory effects of uORFs has potential for broad therapeutic utility. Translational derepression may be an effective method of treating numerous haploinsufficiency disorders inherited in an autosomal dominant pattern. In such disorders, where the cell doesn't produce enough protein due to a loss of function of one allele, derepression of the uORFs in the healthy allele may be used as a mechanism to restore functional protein levels. The existence of disease conditions where the pORF translation is deregulated by uORF loss of function (LOF) or gain of function mutations, provides further support for this concept[1]. uORFs are *cis*-acting elements on the mRNA sequence and as such are amenable to targeting with various oligonucleotide platforms. Recently, two studies have demonstrated increased translation efficiency at the pORF by targeting uORF initiation codons with short steric block chemically modified antisense oligonucleotides (ASOs)[216,217]. This opens up the possibility for the exploration of additional oligonucleotide and genome editing therapeutic platforms, with the goal of targeted gene expression regulation in disease conditions.

The work presented in this thesis demonstrates that uORF effects can be attenuated by functionally silencing the uORF initiation codon. Antisense oligonucleotides evaluated for their ability to interfere with uORF initiation based on steric block and ADAR-mediated RNA editing are discussed in the Results 3 section.

1.4 Oligonucleotide therapeutics

Oligonucleotide (ON) are short sequences of chemically modified nucleotides that use base pairing to cognate or near-cognate target sequence regions to regulate gene expression through several different mechanisms[218-220]. With significant development efforts in the past four decades, the potential of ON therapeutics has recently been validated with several drug

approvals[218]. The recent success of ONs is a result of design and chemical modification discoveries that improve the biological activity of ONs.

1.4.1 Chemical modifications of oligonucleotide therapeutics

Chemical modifications can significantly improve the pharmacokinetic (PK) and pharmacodynamic (PD) parameters of ONs. These parameters may include stability, trafficking, delivery, specificity, efficacy, reduction of off-target effects, and reduction of toxicity. Chemical modifications may have significant effects on the ability of ONs to interact with their nucleic acid target, or the interacting proteins through which they exert their effects. This notion has driven the discovery of thousands of ON chemical modifications, out of which only a fraction has been applied to ONs in the clinic. ONs are typically modified at three positions and include backbone, base, and 2′ modifications. If included, linkers and receptor targeting molecules are typically conjugated at the 5′ or 3′ termini of the ON (**Figure 7**). Chemical modifications have facilitated the conversion of native RNA/DNA molecules which can trigger processes such as mRNA enzymatic degradation or exon skipping *in vitro*, to more drug-like molecules with improved PK/PD parameters. These modifications can influence multiple properties of ONs including charge, lipophilicity, metabolic stability, binding affinity, and efficacy[218,219].

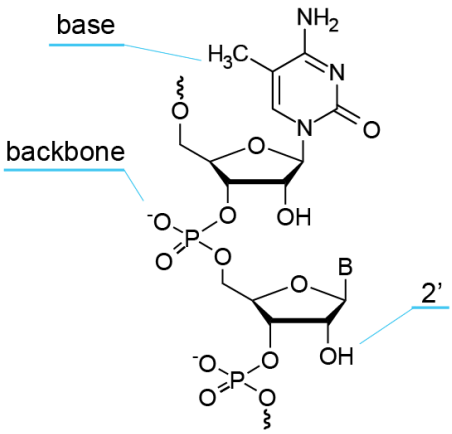


Figure 7. Typical locations of oligonucleotide chemical modifications

A chemical structure representation of two RNA nucleotides connected with a backbone phosphodiester bond. PK/PD parameters of oligonucleotides are improved by the inclusion of chemical modifications, usually included at three sites: backbone, 2′ position of the ribose ring, and base. A 5-Methylcytosine (5meC) base modification is shown. Additional moieties such as linkers and delivery molecules are typically conjugated at the 5′ or 3′ end (not shown).

1.4.1.1 Modifications of the phosphodiester bond

One of the first and most widely-used chemical modification of ONs is the phosphorothioate (PS) backbone modification introduce in pioneering work by Fritz Eckstein[221]. The PS modification replaces an oxygen of the phosphodiester bond (PO) with a sulphur, significantly increasing the stability of the ON by protecting it from nuclease degradation. While originally used to impart nuclease stability, the PS modification has had a major impact on oligonucleotide trafficking and cell uptake[221,222]. The PS modification is compatible for use in RNase H-recruiting ASOs, steric block ASOs, is often featured at the terminal backbone linkages of small interfering RNAs (siRNA), and has been utilized in all approved ON therapeutics to date[223]. The PS modification introduces a chiral center that can exist in an Rp or Sp configuration. Recently developed chemical processes enable the synthesis of chirally pure ONs, where the chirality of the PS modification can be controlled at each

position[224,225]. The potential of chirally pure PS backbone ONs is currently being evaluated in the clinic[226]. However, it can be appreciated that a molecule like Mipomersen which contains 19 stereorandom PS linkages, is a mixture of 2^{19} = 524288 molecules, presumably with differing PK/PD properties. We have demonstrated that chirally pure linkages are advantageous for the stability of ONs, and that certain stereopure motifs can influence the position of the cleavage site of RNase H1[224]. PS Sp linkages were found to increase lipophilicity and impart nucleolytic stability on the oligonucleotide at levels similar to the ones observed in ON gapmers with 2′-*O*-methoxyethyl modified ends[224]. Chiral control of PS modifications may enable protection from nucleolytic degradation by modifying preferred interactions between amino acids of nucleases and prochiral PS centers. A new development for the field is the recently- introduced chirally controlled phosphoryl guanidine-based backbones (PN)[225]. A combination of PS and PN chemistries is currently being evaluated in trials for the treatment of Huntington's disease (HD), amyotropic lateral sclerosis (ALS)/frontotemporal dementia (FTD), and Duchenne muscular dystrophy (DMD)[226]. One potential drawback of the PS modification is the reduction of binding affinity to the target RNA. Binding affinity of ONs can be improved with the addition of 2′ modifications which can influence base stacking and hydrogen bonding distances[223,227].

1.4.1.2 Base modifications

A frequently used base modification is the 5-Methylcytosine (5meC) (**Figure 7**). This modification, naturally found in many organisms including mammals and constitutes a methyl group attached to the carbon at position 5 of the cytosine ring in DNA or RNA. The 5meC is an important endogenous post-transcriptional modification with a functional role in many processes including: (i) epigenetic programming primarily found in CpG-rich regions, (ii) silencing of promoters and enhancers, inactivation or repression of chromatin, and (iii)

suppression of undesired gene expression of retroviruses [228,229]. Aberrant hypermethylation events in the genome have been associated with neurological disease and are prevalent in cancer[230,231]. In ONs, the 5meC modification is used in CG motifs to reduce immunogenicity by avoiding TLR-9 detection, but has also been found to increase binding affinity[232,233].

1.4.1.3 2′ Modifications

Ribose modifications are commonly used to replace the RNA 2′-OH position of the ribose, which can be attacked and hydrolyzed by ribonucleases (**Figure 8**). Namely, during hydrolysis ribonucleases attack and deprotonate the 2′-OH position, leaving a reactive oxygen that mounts a nucleophilic attack on the phosphorus of the phosphodiester bond. This results in an intermediary unstable species where the phosphorus of the phosphodiester bond is bound to five oxygen atoms. The phosphorus detaches from the oxygen connecting it to the sugar resulting in backbone cleavage[234]. This is the reason why RNA is more prone to degradation than DNA and highlights the protective nature of the PS bond and 2′ modifications.

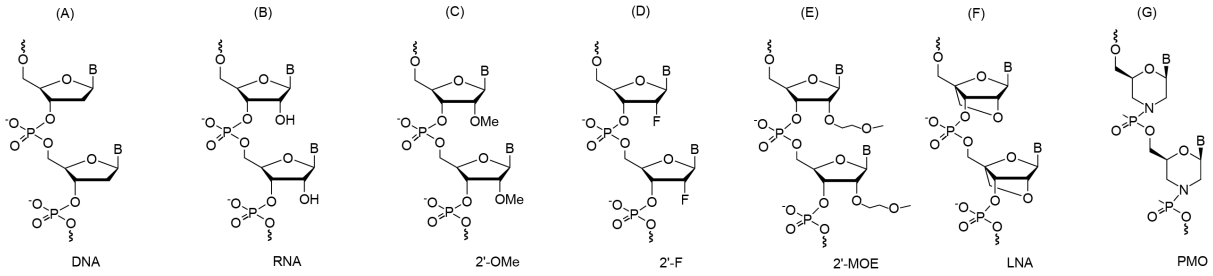


Figure 8. Common ON modifications at the 2′ position of the RNA ribose.

Examples of common chemical modifications at the 2′ position of the RNA ribose. Two nucleotides connected with a nascent phosphodiester bond are shown. (A) Unmodified DNA (B) Unmodified RNA (C) 2′-*O*-methyl (2′-OMe) (D) 2′-Fluoro (E) 2′-*O*-methoxyethyl (2′-O-MOE) (F) Locked nucleic acid (LNA) (G) Phosphorodiamidate morpholino (PMO)

While their original purpose is to stabilize the ON, 2′ modifications have been found to change multiple properties of ONs. For instance, the 2′-*O*-methyl (2′-OMe) modification can improve the ON affinity for the target mRNA and reduce immunogenicity[235,236]. The closely related analog 2′-*O*-methoxyethyl (2′-O-MOE) further improves target affinity. Also known as 2′-MOE, this modification is not suitable for creating RNase H substrates but is used to stabilize the wings of ASO gapmer molecules which typically feature 5 2′-MOE-modified bases on each end and a 10 nt DNA center with a full PS backbone. A notable example is the ASO gapmer Mipomersen, for the treatment of homozygous familial hypercholesterolemia[219,224,237,238]. In addition to this fully 2′-MOE modified ONs are not substrates for RNase H and can be used as steric block oligonucleotides. The spinal muscular atrophy (SMA) drug Nusinersen (Spinraza) and the Batten's disease n-of-1 drug Milasen, use full PS, full 2′-MOE architecture[219,239-241]. The 2′-Fluoro modification significantly increases the affinity toward the target due the highly electronegative fluorine. 2′-Fluoro and 2′-OMe modifications are typically used in siRNAs[219]. Some chemical modifications (locked nucleic acid (LNA), phosphorodiamidate morpholino oligos (PMO)) can also include changes in the sugar ring (**Figure 8F,G**). The locked nucleic acid (LNA) modification, "locks" the ribose into a preferred conformation, increasing the melting temperature (T_m) to ~3-8°C per modified position[218,242,243]. The PMO modification replaces a pentose sugar with a morpholine ring, and the phosphodiester backbone with a neutral phosphorodiamidate (PDA) linkage. PMOs are neutral at physiological pH, which decreases protein binding and improves their toxicity profile. Given that they are not a substrate for RNase H they are primarily used in steric blocking ONs[244]. The PMO modification imparts resistance to nucleases and improves the efficacy and specificity of ONs[245]. PMOs were used in the first FDA approved ONs for the treatment of DMD. Eteplirsen, a 30-nucleotide phosphorodiamidate morpholino oligomer

(PMO) developed by Sarepta Therapeutics, promotes the skipping of exon 51 of the *DMD* gene, and corrects the reading frame[246].

1.4.1.4 ON delivery conjugates

One of the primary reasons to chemically modify ONs is to increase cellular uptake in a particular tissue of interest. The most successful strategy for targeted delivery of ONs has been used for targeting liver disease targets with siRNAs and utilizes a conjugated N-acetylgalactosamine (GalNAc) to target the asialoglycoprotein receptor (ASGPR)[247,248]. Due to the abundant expression of the ASGPR receptor on the surface of hepatocytes, the GalNAc conjugate, in triantennary form, has been demonstrated to be a successful delivery method for siRNAs. Several siRNA drugs using triantennary GalNac delivery have been approved, all delivered subcutaneously and targeting liver targets (**Table 2**)[249-252]. Several additional companies with siRNA platforms, including Arrowhead Pharmaceuticals and Dicerna are testing variants of GalNAc linkers and conjugates in the clinic[247].

Generic Name	Brand Name	Condition
Inclisiran	Leqvio	Hypercholesterolemia
Givosiran	Givlaari	Acute hepatic porphyria
Lumasiran	Oxlumo	Primary hyperoxaluria
Vitusiran	Amvuttra	Polyneuropathy or hereditary transthyretin-mediated amyloidosis

Table 2. FDA-approved siRNA therapeutics featuring a triantennary GalNAc moiety

To achieve optimal activity *in vivo* splice switching steric blocking ONs must cross the cell membrane, but also enter cell nuclei. One novel strategy that holds the promise to enhance delivery of ONs is the conjugation of cell-penetrating peptide ligands (CPP). CPP-conjugated oligonucleotides have been shown by the Gait and Wood laboratories to significantly improve activity of steric block ONs both *in vitro* and *in vivo*[244]. CPPs can be conjugated to PMOs to

create peptide-PMO (PPMO) – a new class of ONs with improved delivery profile in cardiac and skeletal muscles[253]. SRP-5051(vesleteplirsen), an exon skipping PPMO ON developed by Sarepta Therapeutics is currently being evaluated in Phase II clinical trials (NCT04004065). Vesleteplirsen is designed to skip Exon 51 of the dystrophin gene pre-mRNA in order to restore dystrophin levels. The study is currently on hold due to a serious adverse event of hypomagnesemia, but is expected to resume as the hypomagnesemia was transient and could be treated by supplementing magnesium[254].

1.4.2 Modes of oligonucleotide target engagement

ONs can act through a variety of mechanisms which include recruitment of endonucleolytic mRNA degradation by ASOs and siRNA, splice-switching by using steric blocking ONs, and most recently mRNA editing. The following subsections review these mechanisms of actions and clinical their clinical applications.

1.4.2.1 RNase H-recruiting gapmer ASOs

ASO gapmers are single-stranded ONs with a length of ~20 nt where ~5 bases on each side are modified with an RNA-analog chemistry at the 2′ position, while the center uses DNA nucleotides. The first approved gapmer ASO was Mipomersen sodium (Kynamro), a 20 nucleotide ON inhibitor of apolipoprotein B developed by Ionis Pharmaceuticals for the treatment of familial hypercholesterolemia[237,238]. The PO linkages in ASO gapmers are typically replaced with the stabilizing PS backbone. The center region of ASO gapmers creates a DNA/RNA heteroduplex with the target mRNA region and recruits the enzyme RNase H1, which cleaves the mRNA[255]. The cleaved mRNA is no longer a suitable substrate for protein synthesis and is quickly degraded, resulting in a decrease of protein levels. ASOs can be designed to specifically discriminate between two alleles based on a single nucleotide

polymorphism (SNP) difference. One such trial is currently testing allele-specific ASOs containing chirally-pure PS modifications to selectively knock down the repeat expansion mutant allele in patients with Huntington's disease[224,256].

1.4.2.2 RNA interference (RNAi)

The RNAi mechanism is a conserved post-transcriptional gene silencing pathway that has evolved to regulate the expression of protein-coding genes and degrade exogenous nucleic acids[257-259]. The phenomenon of RNAi was first reported in plants by Napoli and Jorgensen, while the double-stranded RNA nature of the mechanism was described by Fire and Mello in the *C.elegans* model[257,260]. Subsequent work by Tuschl and colleagues, revealed that 21-nucleotide double-stranded RNAs can mediate RNAi in mammalian cells, unlocking the potential of RNAi for drug development[257,261].

The mechanism is triggered by small interfering RNAs (siRNAs) – double-stranded non-coding RNAs, with an approximate length of ~20-24 base pairs[261]. While, endogenous siRNAs are processed from long double-stranded RNA (dsRNA) precursors, exogenous siRNA ONs are chemically-modified double-stranded duplexes that may or may not require further processing[219,261]. siRNAs share core aspects of the RNAi pathway with microRNAs (miRNAs), but differ in the mechanism by which they regulate target mRNAs. miRNAs typically exert their effect by incomplete pairing to classes of mRNAs, where siRNAs bind without mismatches and recruit cleavage of the target mRNA; though miRNA repression can also result in mRNA degradation[262,263]. The aim of ON siRNA design is to recruit the endogenous pathway and direct it to knock down specific mRNA targets. Clinical success in the siRNA field has recently reinvigorated the ON drug discovery field, with Alnylam Pharmaceuticals leading innovation with their approved siRNA ONs Patisiran (Onpattro), Inclisiran (Leqvio), Givosiran (Givlaari), Lumasiran (Oxlumo), and Vutisiran (Amvuttra)[249-

[252]. Dicerna pharmaceuticals, and Arrowhead pharmaceuticals are also testing siRNA therapeutics in the clinic[247]. siRNA ONs typically utilize a mixture of 2′-OMe, 2′-Fluoro and PS modifications, coupled with conjugated GalNac for targeted delivery to liver. One exception is Patisiran which uses a lipid nanoparticle (LNP) delivery method[264].

1.4.2.3 Splice-switching steric blocking ONs

Splice-switching ONs have emerged as a promising therapeutic platform for precision interference with the process of splicing. This class of ONs, belongs to the category of steric blocking ONs, are a chemically modified form of ASOs, which does not promote enzymatic degradation of mRNA, but rather blocks the recognition of sequence patterns, to coerce the splicing mechanism to either include or exclude an exon or an intron. More conventionally, splice alteration in the pre-mRNA can be used as a strategy to: (i) remove exons that contain deleterious mutations such as stop codons, or out-of-frame insertions or deletions, or (ii) remove or include exons in order to restore a frame (iii) remove premature codon-containing poison exons to rescue the mRNA from being degraded via NMD[218,220,265,266]. The most notable examples of splice-switching ONs are the now approved therapeutics Nusinersen (Spinraza) for treatment of spinal muscular atrophy (SMA) and Eteplirsen (Exondys 51) for treatment of Duchenne muscular dystrophy (DMD) (Discussed further below)[244].

SMA is an autosomal recessive neurodegenerative disorder caused by a deficiency of the motor neuron (SMN) due to the presence of inactivating mutations in the *SMN1* gene[267]. Fortunately, an evolutionary duplication event has given rise to *SMN2*; a nearly identical copy of the *SMN1* gene[268]. However, due to a C to T mutation at position 6 of exon 7, ~90% of the transcript from the *SMN2* locus results in an unstable SMNΔ7 protein that lacks exon 7[269]. These discoveries mostly came from early work from the laboratory of Adrian Krainer[270-272]. Successful inclusion of exon 7 was achieved with a steric block ASO, as a result of a

collaborative effort by the Krainer laboratory and Ionis pharmaceuticals[273]. Given that the remaining ~10% of the transcript encodes a functional SMN protein, it was proposed that an ASO strategy can be used to shift the transcription balance of the *SMN2* locus towards exon 7 inclusion; increasing the generation of a full length SMN protein. This was eventually achieved by the discovery and blocking of an intron splicing silence N1 (ISS-N1) sequence in intron 7 of the *SMN2* gene with the antisense oligonucleotide (ASO) drug Nusinersen (Spinraza) [274,275].

Another instance of oligonucleotide capable of altering splicing is Eteplirsen (Exondys 51); a morpholino-based ASO for the treatment of DMD. DMD is an inherited neuromuscular disorder characterized by progressive weakness and degeneration of muscle tissue due to an absence of a functional dystrophin protein [276]. This severe form of muscular dystrophy is caused by mutations in the *DMD* gene coding for dystrophin; a membrane-associated protein serving as a structural link between the muscle cytoskeleton and extracellular matrix [277,278]. Unlike in SMA, where the ON is used to promote exon inclusion, Eteplirsen is used to induce skipping of exon 51, aiming to restore the reading frame [279-281]. The skipping of exon 51 is applicable for the treatment of various mutations that can occur in other exons. This includes deletions that can occur to the 5′ or 3′ of exon 51, or mutations in exons 50 and 52 which can be removed[282]. In DMD, the outcome of exon skipping generates a truncated, but functional dystrophin protein, as observed in the less severe: Becker muscular dystrophy(BMD)[283]. BMD is a milder form of dystrophy which is mostly represented by in-frame deletions. Unlike premature termination, or frame shifting mutations, the in-frame deletions in BMD result in partially functional proteoforms with internal truncation[284]. Thus, the exon skipping mechanism used by Eteplirsen generates proteins similar to those observed in BMD.

Splice-switching ONs have also been applied to rescue mRNAs transcripts containing poison exons from degradation. The targeted augmentation of nuclear gene output (TANGO)

technology is designed to splice out poison exons (exons with premature stop codons resulting in non-productive transcripts) in transcripts that are normally degraded via the NMD pathway. The exclusion of the premature termination codon increases the pool of productive transcripts, which in turn leads to elevated protein levels[285]. This technology is currently being evaluated in the clinic for the treatment of Dravet syndrome, a devastating pediatric developmental and epileptic encephalopathy, caused by decreased levels of the voltage-gates sodium channel α subunit $Na_V1.1$, encoded by the *SCN1A* gene[286,287].

Recently, short ~16 nt steric blocking ONs using PO, PS, 2′-*O*-methoxyethyl and 2′-*O*-methyl chemical modifications were reported to be able to interfere with the translation mechanism and alleviate the inhibitory effects of uORFs on the translation of the *RNASEH1* pORF[216,217]. Intriguingly, the upregulation effect was observed even with short PO modified ONs. While this work highlights the possibility of using ONs to upregulate the pORF, it has not been discussed or reproduced in literature since the original publication. (Discussed in Results 3).

1.4.2.4 RNA editing

Adenosine deaminases acting on RNA (ADARs) are a group of enzymes that catalyze the conversion of Adenosine (A) to Inosine (I) in double-stranded RNA duplexes. Inosine can form two hydrogen bonds with cytosine and is therefore recognized by the translational machinery as Guanine (G)[288,289]. In mammals, ADARs are encoded by the IFN-inducible *ADAR1*, *ADAR2* - which are catalytically active, and *ADAR3* - which has not been shown to possess enzymatic activity and is primarily expressed in the brain[290]. Most of the catalytic activity has been attributed to *ADAR1*. *ADAR1* is represented by two isoforms: (i) a constitutively-expressed short p110 isoform, and (ii) an inducible, and longer p150 isoform which is upregulated in response to IFN. The shorter p110 isoform is primarily localized in

the nucleus, while p150 shuttles between the nucleus and cytoplasm due to the presence of a nuclear export signal sequence at its N-terminus[291-293].

The ADAR gene family has been implicated in the regulation of neuronal function, autoimmunity, and may play a critical role in the innate immune response to viral infections[294]. While more research is needed to better understand virus – ADAR interactions, a bias towards A-to-G, and U-to-C mutations has been described in a range of viruses; including SARS-CoV-2[295]. Namely, a recent study suggests that ADAR-induced deamination of RNA is a significant source of mutation in SARS-CoV-2[296]. Conceptually, pervasive A-to-G editing of the viral genome is likely to result in corruption of the message. On the other hand, inefficient editing could also result in advantageous variants for the virus. Therefore, ADARs have the potential to be both anti-viral and pro-viral, and may be important for viral evolution[295].

The key factor in directing the ADAR enzymatic activity to a particular Adenosine is the detection of a double-stranded substrate. The search for the optimal ON trigger architecture and chemistry is still ongoing, as A-to-I editing has been achieved with a very diverse set of ONs including: unmodified long single stranded RNA, double-stranded stem loop RNA, single stranded chemically modified RNA, and circular RNA[297-300]. The common feature of these approaches is that the ON must either create, or include as part of its design, a double-stranded RNA domain to recruit ADAR, and a targeting domain which anneals at the location of the target Adenosine. It has been observed that placing a C mismatch opposite of the editing A improves editing[297,300].

1.4.3 Challenges in oligonucleotide drug development

While recent successes in the clinic have reinvigorated the field of ON drug development, targeted delivery and pharmacokinetics of ONs remains a critical challenge for most human tissues. Before engaging their intended target ONs must overcome significant

extracellular and intracellular challenges. These include: nuclease degradation, scavenging by the reticuloendothelial system, filtration by the kidney, crossing through the capillary endothelium, accessing the tissue interstitium, crossing the cellular membrane, escaping from endosomes, accessing the proper cellular compartment, competing with endogenous molecules for mechanism of action, and finally overcoming these challenges in concentrations that are sufficient to result in a desired phenotypic effect[301].

1.4.3.1 Innate immune defense mechanisms

Due to their nucleic acid character, ONs may activate innate immune system responses evolved to detect self or foreign nucleic acids as signals for cellular damage or infection. These responses are mediated by a diverse group of receptors that are capable of detecting sequence and structural aspects of nucleic acid molecules and can be broadly categorized as DNA or RNA sensors. DNA-sensing immune pathways include the Toll-like receptor 9 (TLR9) pathway which senses non-methylated cytosine-phosphate-guanine (CpG) sequence motifs and interprets their detection as a bacterial or viral infection, and the cyclic GMP-AMP synthase - stimulator of interferon genes (cGAS-STING) pathway which detects foreign or endogenous short DNA molecules and activates the inflammasome[302-304]. RNA molecules are detected by TLR receptors in the endosome (TLR3, TLR7, and TLR8), and the Retinoic acid inducible gene I (RIG-I) receptor which detects blunt-end double-stranded RNAs with a diphosphate or triphosphate at their 5′ ends in the cytoplasm[219,301,302]. Thus, for ONs to be safe, their designs must consider the sequence patterns and structures detected by these cellular defense sensors and avoid their stimulation by applying nucleotide/backbone chemical modification and sequence selection strategies.

1.4.3.2 Chemical modification architectures

Chemical modifications have mostly neutralized severe toxicity effects in many classes of ONs. However, the introduction of chemical modifications can also be a trade-off between improved PK and modification-related toxicity. A more complete understanding of the mechanisms by which chemical modifications augment the ON properties and interact in the cells is critical, as some class chemical modifications toxicity have been reported[305]. Interestingly, a lot of the ON-associated toxicity effects are related to the liver and kidney likely due to the tendency of ONs to accumulate in these tissues in higher concentrations[219,306]. Thus, a balance has to be reached by tuning the types of chemistries and the amount of chemistry included to get an optimal PK/PD profile. However, choosing an optimal chemical architecture is challenging, as millions of chemical modification permutations are possible for any given ON sequence.

1.4.3.3 Internalization, size, and charge

ONs are polyanionic, hydrophilic macromolecules, with sizes exceeding the typical ~500 Da cell membrane diffusion limits of small molecule drugs. For instance, Nusinersen (Spinraza), an approved ON therapeutic for the treatment of SMA (Spinal Muscular Atrophy), has a molecular weight of 7.5 kDa, in comparison to the small molecule drug Aspirin which has a molecular weight of only 179.15 Da[307]. The influence of the size aspect of ONs on delivery is significant, as free cell entry potential and bioavailability of ONs is limited. Given that ON size is not a property that can be reasonably addressed, some chemical modifications (e.g. PMO, PN backbones) have been designed to improve internalization by addressing the negative charge of PS ONs and making them more neutral[225,244].

1.4.3.4 Off-target effects

An additional challenge in developing ON therapeutics is their potential to engage unanticipated targets – also known as off-target effects[308]. Off-target effects can be described as the regulation of unintended targets based on sequence homology driven hybridization. The understanding of off-target effect is a critical process during drug development, with particularly significant implications for the field of genome editing, due to irreversible nature of genome editing platforms[309]. Off-target effects have been clearly associated with toxicity; particularly in targeting paradigms capable of recruiting endonucleolytic mRNA degradation (e.g. siRNA and ASO gapmers)[305,310]. In general, transcriptome analysis shows that dosage of any ON drug may cause expression perturbation in a significant number of genes[311]. These genes may be deregulated for a number of reasons including sequence-specific deregulation, effects arising due to chemical modifications, effects specific to delivery mechanisms (e.g. LNPs or conjugates), or gene regulatory network disruptions, which may perturb the expression of multiple genes in a pathway. Therefore, it is challenging to predict or explain most of those perturbations. Notably, some of those perturbations may be a direct side effect of on-target effects, while others may not necessarily cause unwanted, toxic phenotypes. While more research is required to fully understand the molecular mechanisms that cause off-target effects, critical unanticipated effects have been mitigated by changes to chemical modification architecture, ON length, or targeting position which results in sequence changes in the ON[311,312]. The functional significance of off-target effects depends on modality, and is most critical for the genome editing field, due to the permanent aspect of modifying unanticipated targets.

1.4.3.5 Targeted tissue delivery

Multiple ON siRNA drugs have now been approved. However, recent successes have been limited to CNS and liver[219]. Targeted delivery to other desirable tissues remains a major obstacle and poses a significant challenge in revolutionizing the field of genetics medicines. ON delivery has been pursued through multiple avenues depending on the target tissue and platform[218-220,255]. For siRNAs delivery in the clinic has been successful in formulation with lipid nanoparticles (LNPs) and receptor-targeting conjugates (GalNAc)[313]. From a drug manufacturing perspective, the GalNAc-conjugated siRNAs are a more optimal platform as LNP delivery requires characterizations and manufacturing of additional complex molecules and their subsequent formulation. However, LNP delivery has also been reinvigorated with massive adoption in mRNA vaccine delivery during the Covid-19 pandemic[314]. In contrast to the siRNA mechanism of delivery, splice-switching ASOs and ASO gapmers have typically been delivered intravenously or intrathecally[218,220].

1.4.4 Oligonucleotides as personal medicines

The availability of proven chemistry platforms, coupled with the diverse ability of ONs to regulate gene expression open the possibility for bespoke drug development which can disrupt the biopharma sector. Multiple genetic diseases are rare, but some conditions are so rare that they are yet to be genetically defined. However, it is important to note that rare disease are not rare in the overall population, as it is estimated that 1 in 17 people will be affected by a rare disease at some point in their lives[315]. Recently, the approval of an ON therapeutic for the treatment of a single patient highlighted the potential for portability of some of the chemical modification platforms for the treatment of new conditions. Milasen, a 22 nt 2′-O-MOE, fully PS ON, inspired by the design and mechanism of action of Nusinersen, was approved for the treatment of one child with neuronal ceroid lipofuscinosis 7 (CLN7, form of Batten's disease)

a fatal neurodegenerative disease[316]. Milasen is a splice-switching steric block ON, designed to correct missplicing and restore *MFSD8* expression[317].

The case of Milasen highlights that ONs that utilize the same delivery mechanism and same chemical modifications, may utilize similar delivery regiments and have similar toxicity profiles. While this opens the possibility for the rapid development of ON therapeutics for additional rare disease, it also raises significant regulatory and pricing questions. The final cost of any therapeutic considers development time, development and manufacturing costs, and patient population. ON therapeutics are featured prominently on the list of most expensive approved therapeutics on the market, with Alnylam's: Onpatro, Givlaari, Oxlumo costing ~$450,000, and Biogen/Ionis's Spinraza costing $750,000 for the first year and $375,000 for the following years. Regulation reform is necessary to drive down the cost of the ON class of therapeutics in general, but there should be some incentives for the development of rare disease therapeutics for conditions with small populations[315].

Thesis hypotheses and aims

This thesis describes upstream open reading frames (uORFs) as attractive therapeutic targets for gene regulation through the following aims:

1. Applying *in silico* analyses and machine learning models to reveal the sequence features that guide translational repression by uORFs.
2. Elucidating the molecular mechanisms of uORF translation inhibition and evaluating the therapeutic potential of protein upregulation by modeling uORFs silencing.
3. Evaluating novel steric block and ADAR-recruiting RNA editing ONs for the purpose of uORF dependent protein upregulation.

2 Materials and methods

2.1 Computational analyses

The computational analysis was performed using the Python programming language on a system with 32 Gb RAM, 32 Gb of dedicated GPU, and a 10-core M1 Max Processor. The software tools generated in this thesis are written using the Python and JavaScript programming languages and utilize the HTML markup language for visualization in the browser.

2.2 Cell culture

All the cell lines used in this thesis: human embryonic kidney HEK293T (CRL-3216, ATCC, Manassas, VA, USA), human glioma SF8628 (SCC127, Sigma-Aldrich, St. Louis, MO, USA) , and cervical carcinoma HeLa (CCL-2, ATCC) cells, were maintained using Dulbecco's Modified Eagle Medium (DMEM) containing 10% fetal bovine serum (FBS), 100 U/ml Penicillin, and 100 µg/ml Streptomycin (Pen/Strep) (herein referred to as complete media), in humidified incubators set at a temperature of 37°C, with 5% CO_2. At passage, or experiment seeding time, cells were detached by adding Trypsin-EDTA 0.25% (Thermo Fisher Scientific, Waltham, MA, USA) followed by a 2-3 minute incubation at 37°C. Trypsin-EDTA, was inactivated using complete media without Pen/Strep, and counted using the Countess 3 automated cell counter (Thermo Fisher Scientific).

2.3 RNA extraction

RNA extraction was performed in a 96-well plate format using the SV-96 Total RNA Isolation System (Promega, WI, USA) by following the manufacturer's protocol with the following exceptions: (i) DNase treatment was performed with 35 µl of DNase solution per well, and (ii) DNase inactivation was performed with 300 µl of DNase STOP reagent. This

step was modified to ensure a more efficient DNase step, in order to decrease a potential signal coming from DNA during the Sanger sequencing. Additional measures to reduce the possibility of DNA signal during sequencing is described in the Sanger sequencing primer design strategy section. RNA was eluted in 100 µl in nuclease-free water and stored at -80°C.

2.4 cDNA synthesis

RNA was denatured at 95°C for 5 minutes, followed by a 4°C incubation step, after which the RNA was immediately placed on ice. This step frees the RNA from secondary structures and bound nucleic acids – including antisense oligonucleotides with high melting temperatures. cDNA was generated with the High Capacity RNA-to-cDNA kit (Thermo Fisher Scientific). For a 20 µl reaction volume, 10 µl of 2X RT Buffer was mixed with 1 µl of 20X RT Enzyme, and 9 µl of denatured total RNA in a 96-well plate format. The plate was briefly centrifuged on a tabletop plate centrifuge. The reaction was incubated at 37°C for 1 hour, followed by a 5 minute enzyme deactivation step at 95°C, and a 4°C protocol end storage step. cDNA was stored at -20°C until ready for use.

2.5 Reporter assay vector generation

2.5.1 Generation of the psiCheck dual-luciferase reporter (pDLR) 5′ UTR vector

The psiCheck dual-luciferase reporter vector encodes a human codon optimized *Renilla* luciferase (RLuc) and a Firefly luciferase (FLuc) gene, driven by the simian virus 40 (SV40) and the herpes simplex virus thymidine kinase (HSV-TK) promoters respectively. This vector was modified to include a BsmBI (type IIs) restriction site sequence utilized to insert 5′ UTR of genes-of-interest (GOI) immediately upstream of the RLuc initiation codon. To achieve this, the pDLR backbone was digested by both StuI and DraIII restriction enzymes, the vector was dephosphorylated to prevent re-ligation, and a Gibson assembly (GA) strategy was used to insert a

GOI cloning site, to generate a pDLR 5′ UTR vector. This pDLR 5′ UTR allows the measurement of the effect of a GOI 5′ UTR on the translation of RLuc, using FLuc as an endogenous control (**Figure 9a)**. This resource was designed by Dr. Thomas Roberts and cloned by Dr. Britt Hanson.

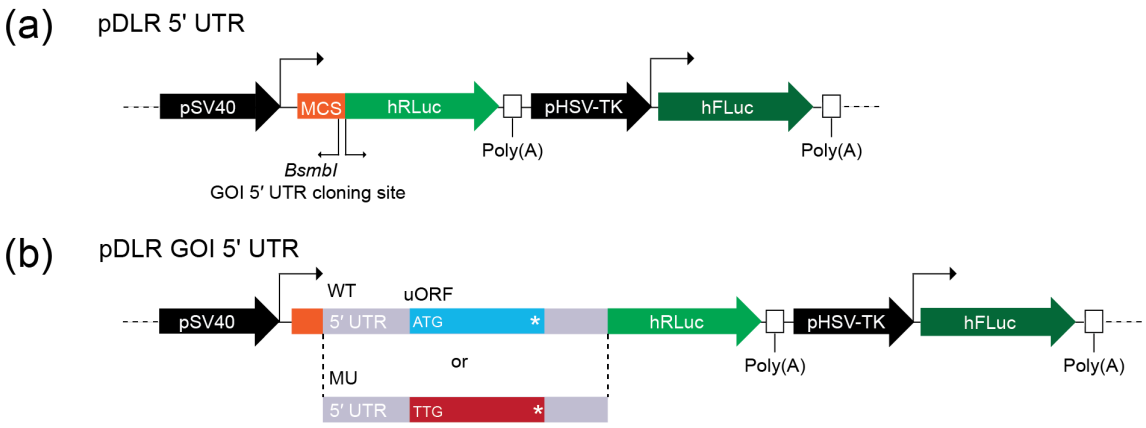


Figure 9. Design of the pDLR 5′ UTR vector and GOI 5′ UTR cloning strategy.

Partial map of the pDLR 5′ UTR vector showing (a) the GOI cloning site immediately before hRLuc, and (b) an example GOI 5′ UTR insert sequence containing a uORF in its wild type ATG initiation codon, or a mutated TTG initiation codon. The clones insert effectively represents a 5′ UTR sequence for the hRLuc pORF, while hFLuc serves as an internal normalization control.

2.5.2 Reporter assay insert design and generation of the pDLR GOI 5′ UTR plasmids

Wild type or mutant 5′ UTR inserts containing Gibson assembly (GA) overhangs were designed *in silico*, and GA cloning was simulated using the SnapGene software (Dotmatics, Boston, MA, USA). Gene blocks were synthesized by Integrated DNA Technologies (IDT, Coralville, IA, USA) or Genewiz (Azenta, Chelmsford, MA, USA). All gene blocks were cloned into the pDLR 5′ UTR vector, which was first digested with BsmBI, followed by a dephosphorylation to prevent re-ligation. GA was used to clone 5′ UTRs of transcripts of interest into the pDLR 5′ UTR. Linearized vectors were incubated with custom gene block sequences containing the 5′ UTR of GOI and 20 bp GA overhangs. To test the effects of uORFs

on translation, the initiation codons were altered from an ATG wild-type (WT) to a TTG (mutant) sequence, putting the uORF in an unfavorable initiation context. Single point mutations were generated using site directed mutagenesis (SDM) with the Q5 site-directed mutagenesis kit (New England Biolabs (NEB), Ipswitch, MA, USA), while a gene block approach was used for more complex 5′ UTR, which included multiple uORFs. The generation of pDLR GOI 5′ UTR places RLuc under translational control of the cloned 5′ UTR sequence (**Figure 9b)**. Plasmids were generated in the Wood/Roberts laboratory, or as a service by Genewiz (Azenta).

2.5.3 Generation of the pDLR HOXA11 5′ UTR uORF vector

The pDLR *HOXA11* 5′ UTR uORF vector uses the pDLR GOI 5′ UTR vector with a cloned 5′ UTR of the *HOXA11* gene (**Figure 25a**). The *HOXA11* gene has one active uORF which can significantly repress expression of the RLuc pORF (**Figure 25c**). In pDLR *HOXA11* 5′ UTR uORF, the *HOXA11* uORF is replaced with a Golden Gate cloning site using two BsmBI sites, allowing for the cloning of changing the uORF sequence and Kozak context, in a *HOXA11* 5′ UTR background. This vector was developed by Dr. Britt Hanson.

2.6 Transfection of Dual-Glo luciferase 5′ UTR model plasmids 24 hours

Minipreps of approximately 4 µg were diluted in 300 µl of nuclease-free water. Plasmid concentration was measured by Nanodrop. Plasmid source transfection plates were prepared at a 50 ng/µl concentration. One day before transfection, HEK293T cells were seeded in opaque 96-well tissue culture plates, in Pen/Strep–containing DMEM at a cell density of 3×10^4 per well and allowed to attach overnight. The following day, the media was replaced with 100 µl of DMEM without Pen/Strep. Lipofectamine 2000 (ThermoFisher, Waltham, MA) – plasmid transfection reagent AB was prepared by mixing 10 µl of solution A (8 µl OptiMEM

(ThermoFisher, Waltham, MA) + 2 µl of 5 ng/µl plasmid DNA) with 10 µl of Solution B (9.5 µl OptiMEM + 0.5 µl Lipofectamine 2000) and incubating at room temperature for 15 minutes to enable transfection complex formation. Forward transfection was performed by adding 20 µl of the transfection reagent AB to each well containing 100 µl of media. Plates were incubated for 24 hours in a humidified incubator set at a temperature of 37°C, with 5% CO_2 before further processing.

2.7 Dual-Glo luciferase (DLR) assay

All plasmids cloned in the pDLR 5′ UTR vector were transfected into HEK293T cells. Transfection volumes were typically 120 µl in opaque tissue culture plates. The DLR assay was performed using the Dual-Glo Luciferase Assay System (Promega, E2980). Prior to plate processing assay buffers were left to equilibrate at room temperature overnight to remove precipitate. Media volumes were gently removed and adjusted to ~75 µl, to make room for the DLR reagents. After room temperature equilibration, 75 µl of Dual-Glo Luciferase Assay reagent were added to 75 µl of media and mixed gently by pipetting up and down. Following a 10 minute incubation, cells were lysed and firefly luminescence was measured on a Spectramax M5 instrument (Molecular Devices, San Jose, CA, USA). After measuring firefly luminescence signal, 75 µl of freshly prepared Dual-Glo Stop & Glo reagent was added to the plate and gently mixed without introducing bubbles. The plate was left to incubate at room temperature for at least 10 minutes. Following incubation, Renilla luminescence was measured. The Renilla/Firefly ratios were calculated and normalized to the ratio of the plasmid control wells.

3 Results 1: Bioinformatics analysis of uORFs

3.1 Introduction

Rapid advancements in high-throughput mass spectrometry and sequencing technologies have resulted in improved resolution datasets of the human genome, proteome, and transcriptome. Scientists have used these datasets to better understand the quantitative relationship between the transcriptome and the proteome. Surprisingly, there is no consensus with regards to what extent mRNA levels can explain protein levels in the cell, and protein levels are often found to not correlate well with transcript levels[126,146,147]. Most studies focusing on understanding the relationship between the transcriptome and the proteome agree that protein levels are highly regulated by post-transcriptional *cis*-acting and *trans*-acting regulatory mechanisms[126,131,132]. A key *cis*-acting regulatory element of translation is the upstream open reading frame (uORF) – a short structural analogue of the primary ORF (pORF), defined by an initiation codon found in the 5′-UTR and an in-frame termination codon, which is generally found in the UTR, but more rarely can be found in the coding sequence (CDS) of the pORF. Multiple studies have linked the presence of uORFs in the 5′ UTR with translational repression of pORF[161,162,166,167,318]. However, some uORF 5′ UTR architectures have been shown to promote translation of the pORF under stress conditions[169-171,173,204]. Deregulation of uORF-mediated translational control has been associated with pathogenic phenotypes[1,177,318].

uORFs are highly represented in the transcriptome, and are represented in 59.4% of the transcriptome, in a single or multi-uORF 5′ UTR architectures. Despite their prevalence, remarkably very few uORFs have been characterized[318]. Ribosome profiling techniques paired with next generation sequencing have revealed a pervasive translation in the 5′ UTR[178,319]. With the advent of technologies such as genome editing, and the rapid advancements in the

oligonucleotide therapeutics field, it is critical to fully characterize the functional mechanisms of uORFs, as they may enable new avenues for therapeutic development.

Multiple uORF features have been implicated in driving uORF activity. Most notably, increased inhibition of the pORF has been associated with a strong uAUG context, long distance from the cap, presence of multiple uORFs, and increased conservation[162,166]. Recently published high-fidelity linked transcriptome and proteome datasets can power more comprehensive uORF feature analyses[131,320]. The study of uORFs necessitates the development of tools that can annotate uORFs in the context of biological data. The ability to generate omics data at scale has also enabled the application of artificial intelligence in biology. Machine learning techniques may be a promising asset for the prediction and understanding of uORF features. The key aim of the research presented in this chapter is to develop tools that facilitate the study of uORFs, characterize uORFs in context of large-scale proteomics and transcriptomics data, and use machine learning to test uORF research hypotheses and gain novel insights, with the ultimate goal of evaluating the therapeutic potential of uORFs.

3.2 Hypothesis and Aims

I hypothesize that the translation repression by uORFs is influenced by the synergistic effects of specific sequence features which can be modeled and predicted in silico.

This hypothesis was tested by:

- Prediction and mapping of all uORFs in the human transcriptome and their respective properties – creating the uORFeome: a comprehensive database of uORFs.
- Creating uORF Detective: a visualization tool for studying uORFs in any transcript in context of ribosome profiling data.
- Examining the presence of uORFs in context of NGS data from 32 human tissues from two of the most recent linked transcriptome/proteome linked studies.
- Using large-scale protein expression datasets to evaluate the repressive potential of multiple uORF features.
- Using machine learning models to discover novel sequence features that may influence uORF activity.

3.3 Materials and Methods

3.3.1 Prediction and mapping of uORFs in the human transcriptome (uORFeome)

The uORFeome is a set of software algorithms that was developed to extract publicly available gene and transcript annotations, to generate a complete database of uORFs and their properties. The source code for the uORFeome was written using the Python programming language. The output of the uORFeome is in a database format. Validation for the measurements of the uORFeome features was performed by manually or programmatic assessment of representative validation gene subsets. Coordinate calculations were validated by cross-referencing to publicly available annotations including the UCSC genome browser database, the NCBI GenBank database, and the GWIPS-viz UCSC instance.

3.3.2 Processing of uORF feature sets

Custom algorithms were developed to create Python Sets (a container with unique elements) based on various uORF features including: uORF length, termination codon, number of uORFs, spanning CDS, uORF position, Kozak context, distance from pORF, frame, stop codon usage. These sets were intersected with transcriptomics and proteomics data from the Wang *et al.* and Jiang *et al.* datasets to generate the cumulative distribution function (CDF) plots[131,132]. Significance was evaluated by calculating a *P*-value using the Kolmogorov-Smirnov (KS) test. However, the provided code also allows for statistical analysis using the Mann-Whitney test. The null hypothesis of the KS test is that the two compared groups have the same distribution.

3.3.3 Processing of the proteomics data

The Wang *et al.* dataset was analyzed in context of uORF properties and their effects on translation. To establish a measure of protein expression, the median expression value across 29 tissues was used[132]. The global analysis was also performed on each tissue separately with similar outcomes.

3.3.4 A strategy for generating unique genomic uORF identifiers

uORFs can be studied on a transcript or gene level. For most splice variants with a common 5′ UTR, an identifier assembled from the chromosome or HGNC gene symbol, and the genomic start and stop coordinates of each uORF may be a viable strategy to describe common uORFs across variants with a single identification. However, this approach is compromised when an exon internal to a uORF is spliced. For instance, in a situation where the uORF start codon is in exon 1 and the uORF stop codon is in exon 3, a variant with a skipped exon 2 will result in a shorter uORF sequence, while a variant with an included exon 2 will result in a longer uORF sequence - resulting in two distinct uORFs. Despite their sequence length differences on a transcript level, the genomic coordinates of these uORFs are identical (**Figure 10**). Therefore, a unique genomic identifier for a uORF must contain additional information such as the actual uORF sequence.

uORF nucleotide sequences can be too long to be used as part of an identifier. By using string hashing, it was possible to represent the uORF sequence with a unique 6 letter alpha-numeric code. For this purpose, the Shake-256 algorithm was used, which is part of the SHA-3 (Secure Hash Algorithms – 3) family of algorithms[321]. Conceptually this employs a strategy like the one used for password security, where a human-readable password is represented with a scrambled alpha-numeric code of a certain length. Theoretically, a sequence with the same set of characters results in the same hash code. For instance, the minimal uORF sequences A

= “AUGUGA”, and B = “AUGUGA”, share the hashed code: “eb4ecb”. More generally, if A=B, then hash(A) = hash(B). Given that multiple genes may share a uORF with the same sequence, adding the gene name, chromosome, and genomic coordinates of the uORF to the input sequence of the hash algorithm ensures uniqueness from other genomic regions. By using this approach, uORFs of transcript variants with different transcription starts are treated as being the same. The 6-character alpha-numeric representation allows for 36^6 possibilities. As such, while “collision” cases where different uORFs may produce the same hash code are theoretically possible, they are not likely to occur. The hashed string length is a tunable parameter which can be further increased. Finally, while the concept of hashing is one-directional, all the information of the hashed sequence is captured so it can be associated with the hash.

This hashing strategy enables ID tagging and mapping of all unique uORFs to the human genome, which are typically repeated across different isoforms. Ultimately, this strategy enables the assessment of the number of all unique uORFs in the genome, and the creation of BED files which can be mapped and visualized in various genome visualization applications.

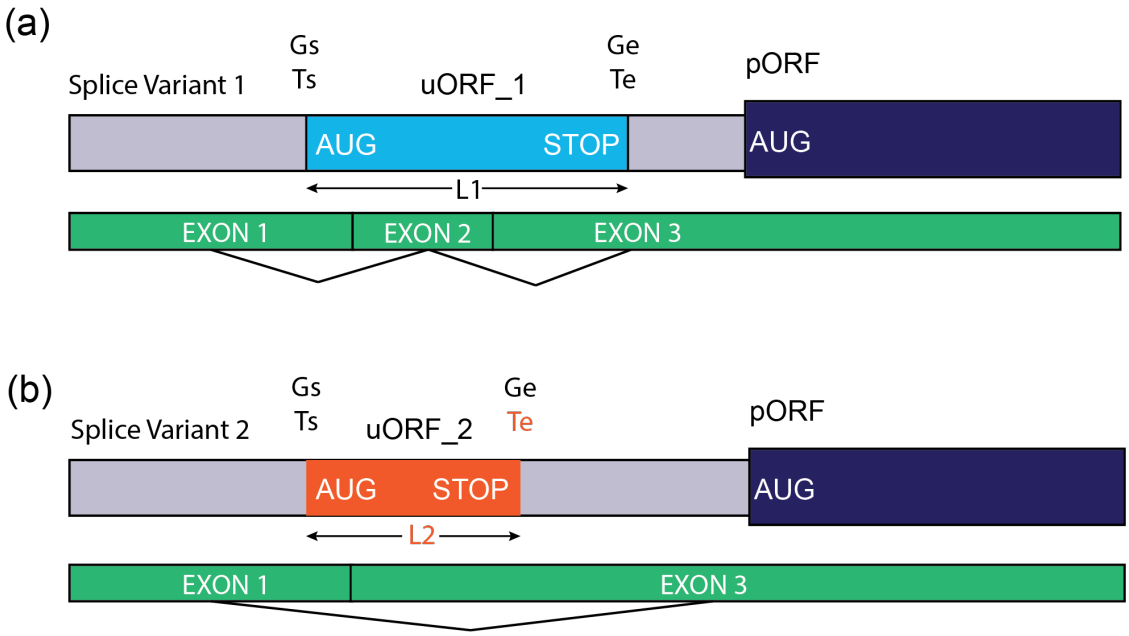


Figure 10. Strategy for generating unique genomic uORF identifiers

A schematic representation of the 5′ UTRs of two splice variants of the same gene containing a single uORF which has identical genomic start (Gs) and genomic end (Ge) coordinates on a genome level but has a different sequence and different transcript end (Te) on a transcript level. Corresponding exons are shown in green, while pORF CDS start is shown in dark blue (a) Splice variant 1, uORF 1 has genomic start and end coordinates (Gs, Ge), transcript start, end coordinates (Ts, Te), and a uORF sequence length of L1. (b) Splice variant 2, due to exon skipping of exon 2, uORF 2 shares the genomic coordinates and the transcript start coordinate with uORF 1, but acquires a new transcript end (Te), and has a different sequence with a shorter length L2.

3.4 Results

3.4.1 uORFeome a database of predicted uORFs and their properties in the human transcriptome

Technological advancements in the field of sequencing continue to improve the accuracy of the transcriptome. uORF properties have been surveyed in previous analyses, but the computational algorithms were either not current or difficult to deploy[162,166,168]. Custom algorithms were developed using the R and Python programming languages to analyze uORF properties on the most recent version of the RefSeq database. The summary of uORF properties in **Figure 11**, represents the basis of the uORFeome – a comprehensive database of uORFs and their properties which was developed for the purposes of this research.

Consistent with previous analyses 59.3% of all human and 48.5% of all mouse transcripts contain 1 or more uORFs in their UTR (**Figure 11a,b**). While the transcriptome has now been updated to 62415 transcripts, the percentage of uORFs has remained relatively unchanged at 59.4% (**Figure 12**). Given that most uORFs are enclosed in the 5′ UTR, they tend to have short lengths and can appear at any position in the 5′ UTR (**Figure 11c,f**). Approximately 20% of all uORFs initiate in the 5′ UTR but terminate in the CDS of the pORF (**Figure 11d**). These uORFs are predicted to be inhibitory as any subsequent reinitiation is no longer productive in context of the pORF translation. uORFs share similar termination codon usage with pORFs (**Figure 11g**). uORFs initiation codons are generally found with weaker Kozak context sequence than the pORF and are conserved. It should be noted that a significant portion of the pORFs have a weak initiation context (**Figure 11h,i**). A database was generated in BED format, mapping the coordinates of uORF for every transcript to Hg38; the *SMN2* uORF visualized in the UCSC genome browser (**Figure 11k**).

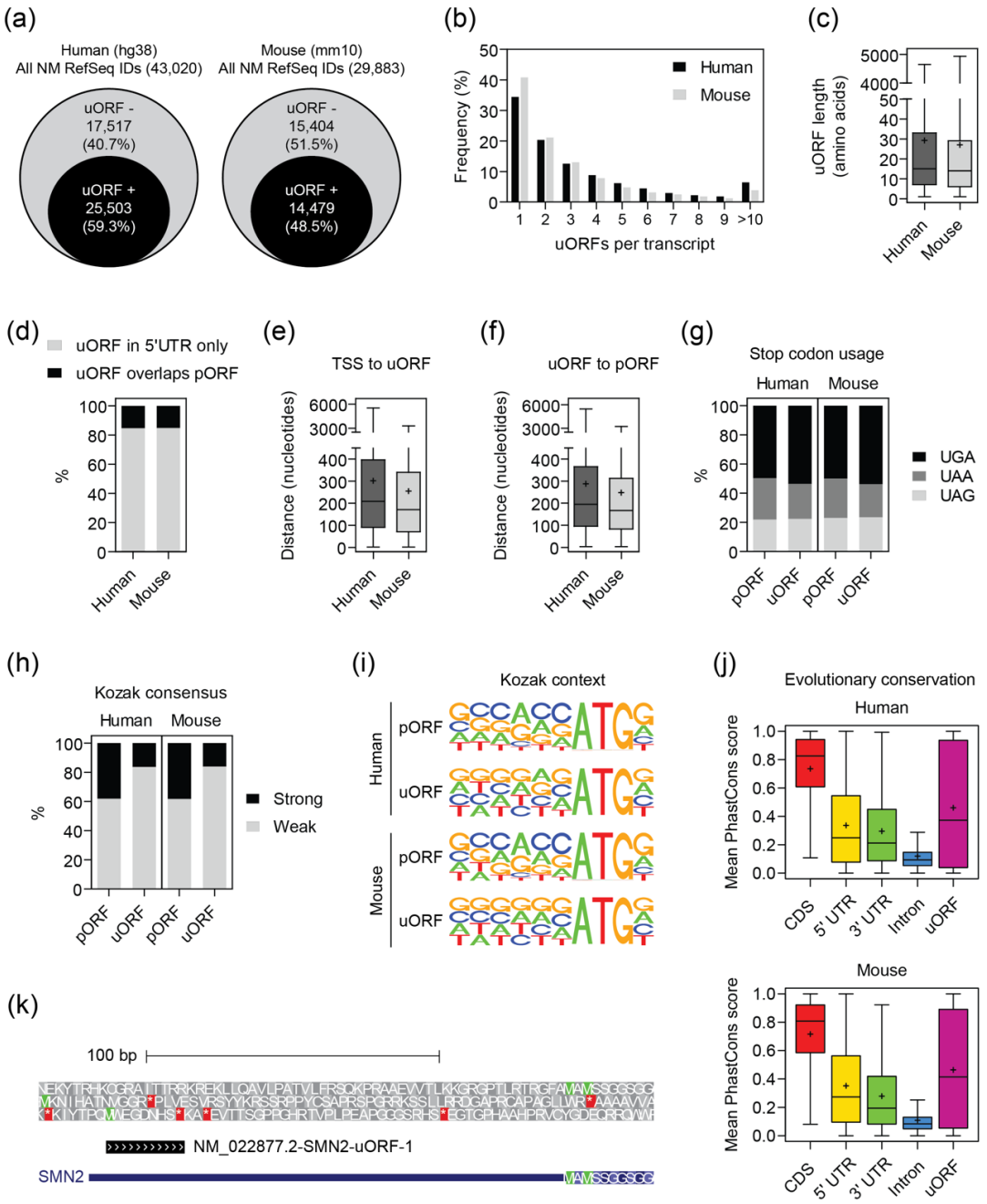


Figure 11. Bioinformatics analysis of uORFs properties in human and mouse

A computational analysis of uORFs in the human and mouse transcriptome comparing uORF representation, homology, and key uORF features (a) uORF prevalence in human and mouse transcripts. (b) uORF frequency. (c) uORF length distribution. (d) uORFs overlapping pORF start. (e) uORF distance from UTR start. (f) uORF distance from pORF initiation. (g) Stop codon usage. (h, i) Kozak sequence context. (j) Conservation. (k) Example of a uORF (black) in the *SMN2* 5′ UTR (blue line). Analysis and figure by Dr. Thomas Roberts, uORFeome was modified and coded in Python by Nenad Svrzikapa.

A set of bioinformatics programs were developed using the Python programing language to generate the uORFeome database. The uORFeome database annotates all human transcripts using real-time and up-to-date information from the UCSC genome browser database. These transcripts are then analyzed with a custom set of algorithms to detect and annotate all uORFs. Once mapped, secondary algorithms analyze and calculate various uORF features including: uORF length, initiation codon, stop codon, Kozak consensus strength and sequence, distance to TIS, distance from the cap, translated sequence etc. A third set of algorithms remap the genomic level ribosome profiling data on the transcriptome. Finally, the uORFeome database links to disease databases such as OMIM and ClinGen, to extract valuable information with regards to disease phenotypes. A recent statistical survey of the uORFeome reveals that 37075 (59.4%) of all human transcripts contain 1 or more AUG uORFs in their UTR sequence (**Figure 12**). The AUG initiation codon occurs at lower frequency in the 5′ UTR when compared to other NCCs (**Figure 12**). However, codon frequency is not the only factor in determining efficiency. A recent study evaluating the efficiency of NCCs in humans shows that in some instances NCC initiation can generate protein expression as high as ~50% of the levels generated by initiating at an AUG codon found in a strong Kozak sequence context[322]. The CUG NCC was found to be the most efficient; followed by ACG and GUG. The AUA, AUC, and UUG codons were found to be less efficient, but produce detectable expression levels[322].

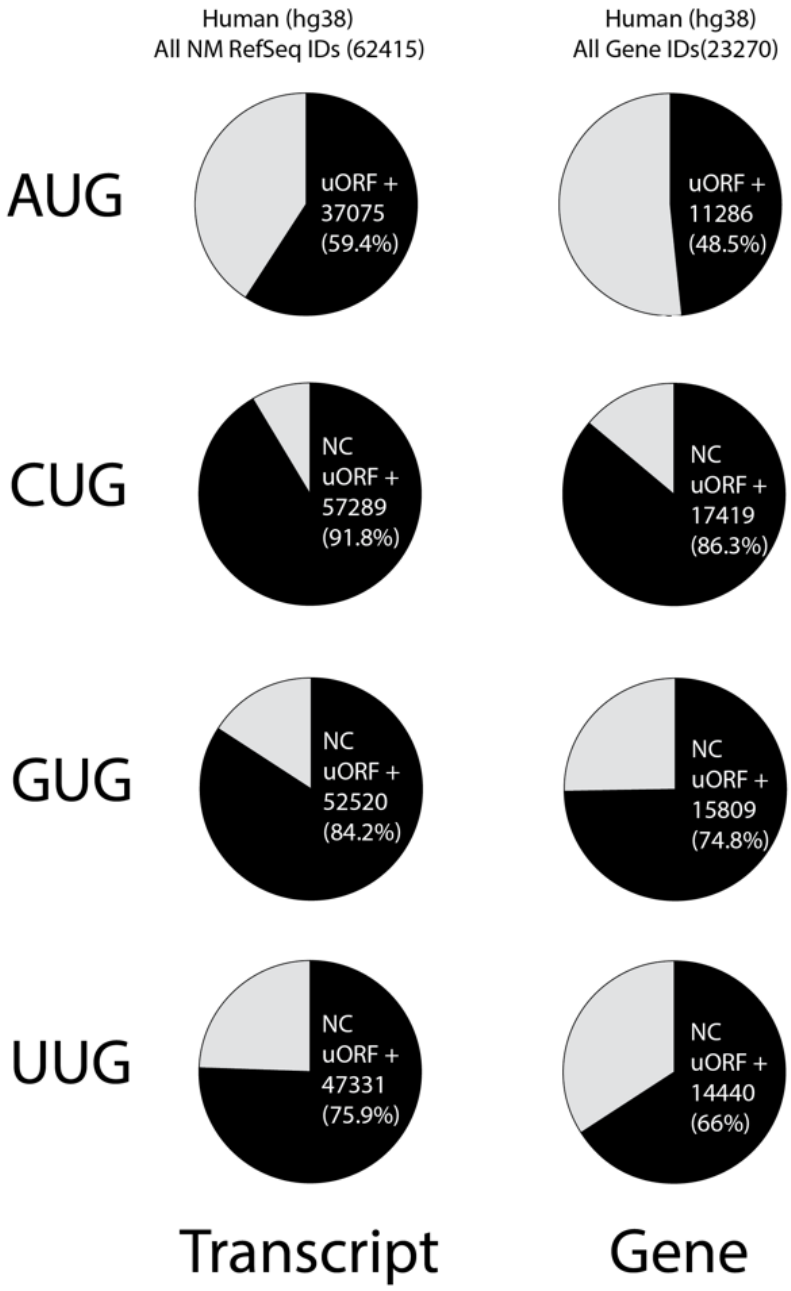


Figure 12. A current uORF analysis on the uORFeome database (Analyzed May 2022)

A computational annotation of uORFs on the human transcriptome and human genome (hg38) with canonical (AUG) and non-canonical (NC) initiation codons reveals the ~60% of all human transcripts have AUG-initiating uORFs in their 5′ UTR, while NC initiation codons are even more prevalent. uORF containing genes or transcripts are denoted as uORF + (black), while the grey subset lacks uORFs.

3.4.2 Probing the effects of uORFs on translation in matched transcriptomics and proteomics datasets

Published bioinformatics analyses of uORFs have relied on datasets with a limited size and may be underpowered. Furthermore, the most notable uORF bioinformatics analysis was performed on mouse proteomics data[162]. The analysis presented here takes advantage of a published quantitative paired proteome and transcriptome atlas measured across 29 healthy human tissues; referred to as the Wang *et al.* dataset[320]. This enables the study of 18,072 transcripts and 13,640 proteins; a significantly larger set of data than previously analyzed with respect to uORF function. To supplement this analysis and confirm findings an additional analysis was performed on the more recent quantitative proteome map of the human body which also includes matched transcriptome data; referred to as the Jiang *et al.* dataset[131]. This resource enables the study of matched mRNA and protein data in 12,627 genes generated from the same tissue samples.

While there is significant overlap in the findings between the Wang *et al.* and Jiang *et al.* studies, the differences in the used experimental and analytical techniques may account for some of the variability in their findings[131,132]. Notably, the Wang *et al.* uses a single sample for each analyzed tissue, which may prevent generalization of some of the findings[131]. In contrast, the Jiang *et al.* study uses 201 samples from the GTeX resource, representing 32 human tissues from 14 healthy individuals. The transcriptome data in Wang *et al.* is represented as fragments per kilobase million (FPKM), while Jiang *et al.* uses the transcripts per kilobase million (TPM) measure which allows for a better comparison of expression between different samples; in line with the study using multiple samples. Most notably, the Wang *et al.* uses a label-free mass spectrometry analysis to generate their proteomics resource, while Jiang *et al.* relies on tandem mass tag (TMT) 10plex/MS3 mass spectrometry strategy[323].

3.4.2.1 uORFs are less represented in highly expressed human transcripts across tissues

The Wang *et al.* and Jiang *et al.* datasets include comprehensive transcriptome data on 29 and 32 human healthy tissues respectively. One of the uORFeome analysis outputs was the set of human genes where all isoforms include uORFs. To examine the distribution of uORFs across various levels of transcript expression, transcriptome data were binned in 100 equally sized bins with the lowest numbered bins representing genes with low mRNA expression as represented by either a FPKM or TPM measure, as appropriate. Prior to this analysis transcripts with FPKM or TPM < 0.5 were filtered out as this is typically considered expression at or lower than background. Next, for each bin, the gene sets at each percentile expression were intersected with the set of known uORF genes to calculate the percentage of uORF-containing genes in the bin. Strikingly, the top ~10% most highly expressed transcripts have a tendency to lack uORFs (**Figure 13A**). While this phenomenon has been previously observed using microarray datasets, this is the first uORF tissue-level analysis that employs high-quality NGS datasets and presents this phenomenon in high resolution [40,162]. This result was further corroborated by performing analysis on the more recent Jiang *et al.* dataset which uses TPM as a measure of mRNA abundance and includes several additional tissues (**Figure 13B**).

These results reveal that transcripts with lower overall abundance are more likely to contain uORFs, suggesting that a large population of transcripts may be regulated by uORFs via the NMD pathway. This is consistent with previous estimates that ~35-50% of transcripts may be susceptible to uORF-mediated NMD. In contrast, the highly expressed transcripts appear to be under high evolutionary pressure to avoid both uORF-regulated degradation by NMD and translation inhibition.

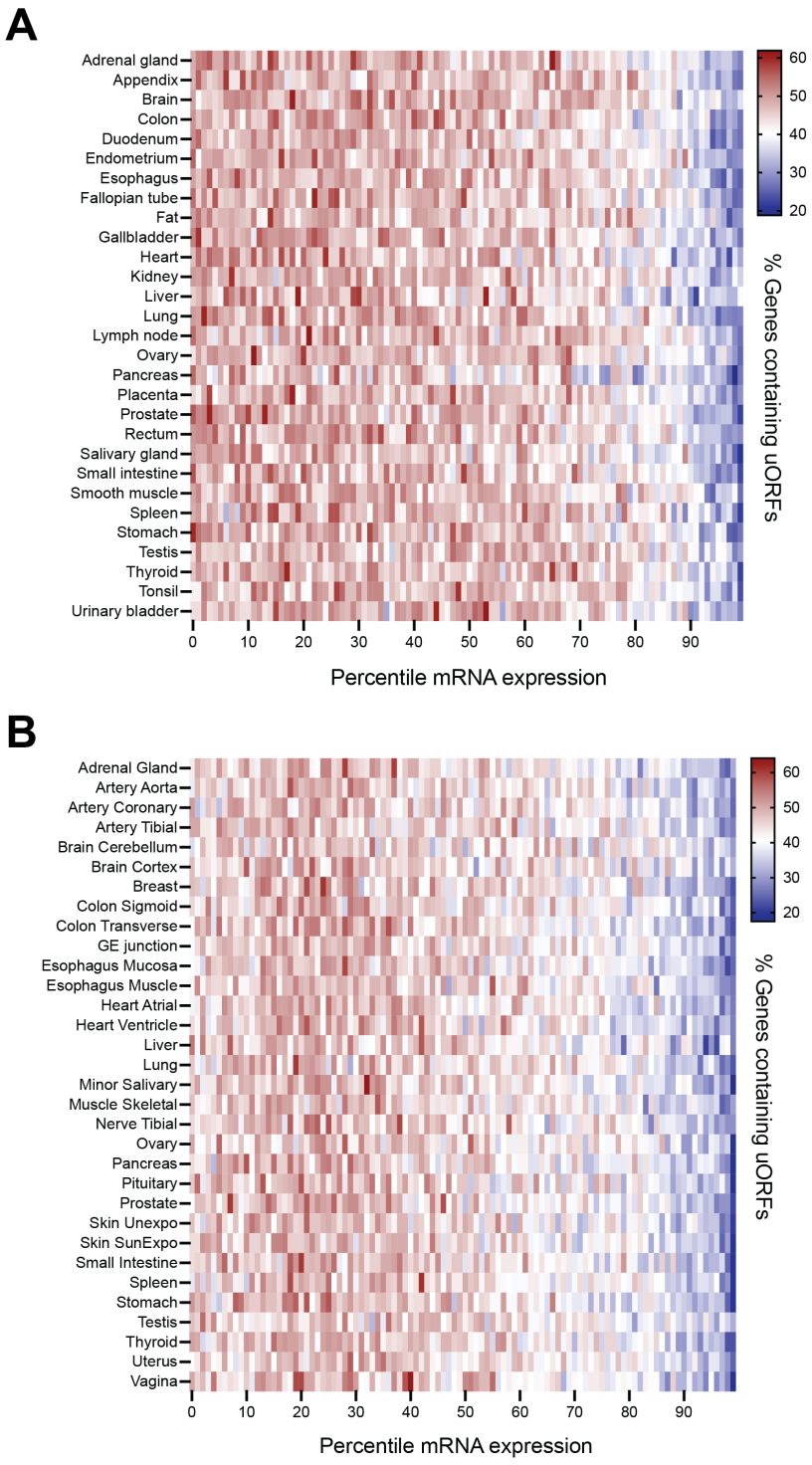


Figure 13. Heatmap of uORF frequency across mRNA expression levels.

A computational analysis of mRNA expression datasets from two studies. The presence of uORFs is evaluated in 100 bins of equal size, ordered based on mRNA expression level (X-axis). (A) Analysis of the Wang *et al.* dataset in 29 human tissues (B) Analysis of the Jiang *et al.* dataset containing 32

tissues. Both analyses reveal tissue independent depletion of uORFs in highly expressed transcripts (blue). Transcripts < 0.5 FPKM or TPM were filtered out.

3.4.2.2 Filtered genes are not enriched in uORFs

A common process in NGS analytical pipelines is the exclusion of data which has lower than 0.5 FPKM or TPM abundance. The reason for this exclusion is the notion that there is not enough data to call the expression level of the transcript with confidence as it is not represented with an above background number of reads. Given that, NMD is one of the regulatory features of uORFs, the transcriptome analysis was also performed without excluding this subset of genes. It was hypothesized that this population may be under stronger NMD control, and may be enriched for uORF presence. However, no uORF enrichment was observed in the < 0.5 FPKM and TPM populations (**Figure 14A, B**). In fact, in the Jiang *et al.* dataset this population appears to be slightly depleted in uORFs (**Figure 14B**)

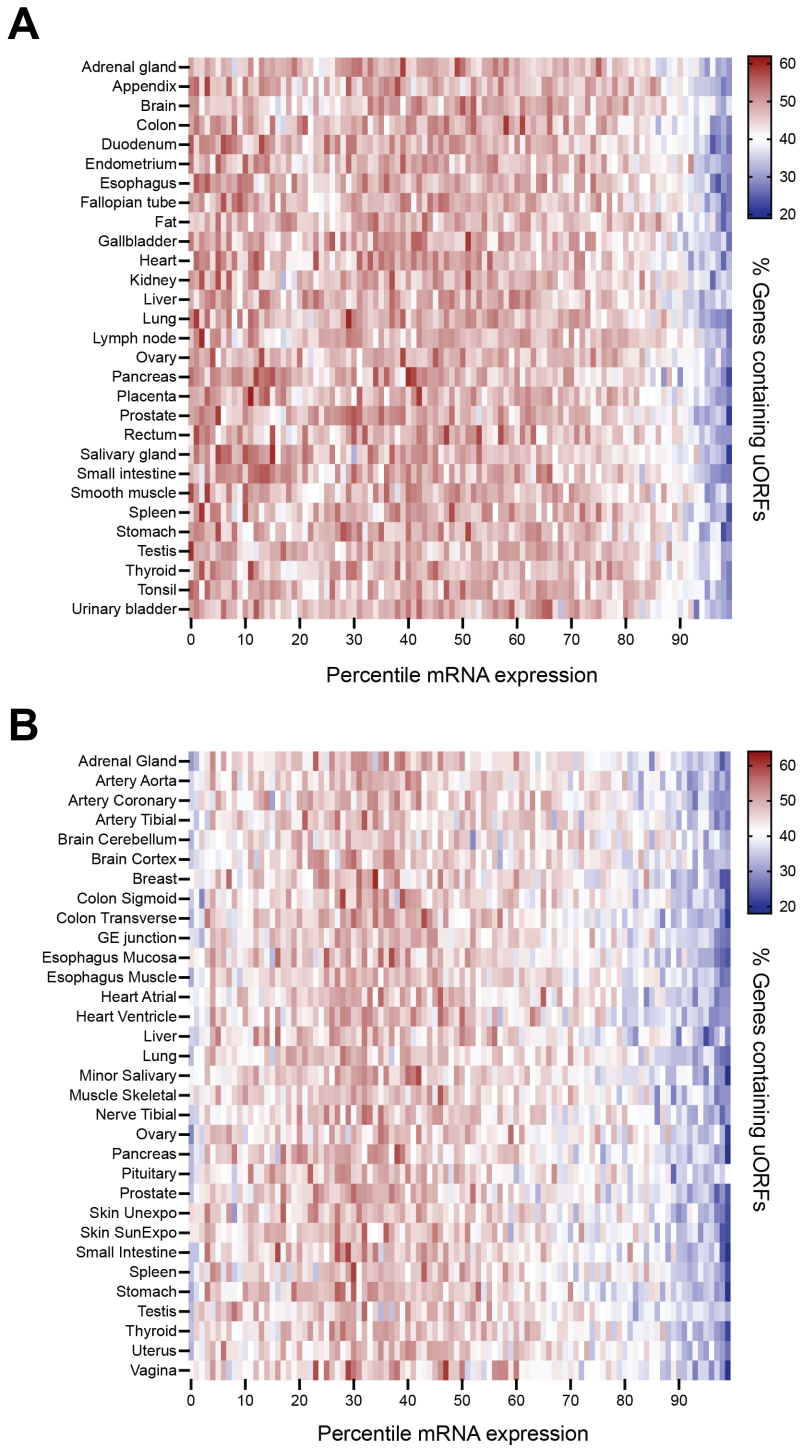


Figure 14. Heatmap of uORF frequency across mRNA expression levels (Unfiltered).

A computational analysis of mRNA expression datasets from two studies. The presence of uORFs is evaluated in 100 bins of equal size, ordered based on mRNA expression level (X-axis). (A) Analysis of the Wang *et al.* dataset in 29 human tissues (B) Analysis of the Jiang *et al.* dataset containing 32 tissues. Both analyses reveal tissue independent depletion of uORFs in highly expressed transcripts (blue). Transcripts < 0.5 FPKM or TPM were filtered out.

3.4.2.3 Gene Ontology (GO) term enrichment analysis of the highly expressed transcript population

As previously mentioned, the ~10% of most highly expressed transcripts appear to be under strong selection to avoid uORF-regulated inhibition. Given that this observation was consistent across both high-throughput NGS datasets, GO term enrichment analysis was performed on the union of the genes present in this subset in the < 0.5 FPKM filtered Wang *et al.* dataset, GO term annotation and enrichment analysis was performed using the GOnet tool[324]. This analysis allowed the mapping of genes to their GO subsets and revealed that the genes in this group are associated in critical cell processes including: cell metabolism, cell response, cell trafficking and translation (*P*-value < 1.76e-6).

3.4.3 Bioinformatics analysis of uORF features and their effects on protein translation

The uORFeome provides a set of measurable uORF features including: uORF position, uORF sequence, translation of sequence, stop codon usage, distance to TIS, Kozak sequence etc. It is unknown to what extent these features influence the regulatory properties of uORFs. Previous computational analysis on uORF features have used mouse proteomics datasets, with limited sample sizes. Recently published human protein atlas datasets can power a much more comprehensive analysis of uORF features and their effects on protein translation[320]. The effects of uORF features on protein expression was achieved by the comparison of the protein expression data of different subsets of genes, belonging to different feature categories.

Several strategies were used to summarize the proteomics data. Data were aggregated across all tissues, analysis was performed on the aggregated data and repeated in each tissue separately with similar results. Cumulative distribution function (CDF) plots were used to evaluate if genes with certain uORF features segregate as statistically different distributions. The nonparametric Kolmogorov-Smirnov (KS) statistical test was used to assess if the analyzed

data belong to the same, or different distribution. As expected, not all proteins are ubiquitously expressed across all tissues. In order to increase the number of proteins that can be analyzed, the median expression value was taken across 29 tissues. To ensure that no bias was introduced by this processing step, the analysis was also performed on each of the 29 tissues separately. It was observed that genes that contain uORFs in their 5′ UTR sequence have a significantly lower median protein expression levels than genes lacking uORFs in both the ubiquitously expressed (*D*=0.15, *P*=5.11e-22) (**Figure 15a**), and the complete dataset proteomics dataset (*D*=0.14, *P*=5.78e-50) (**Figure 15b**).

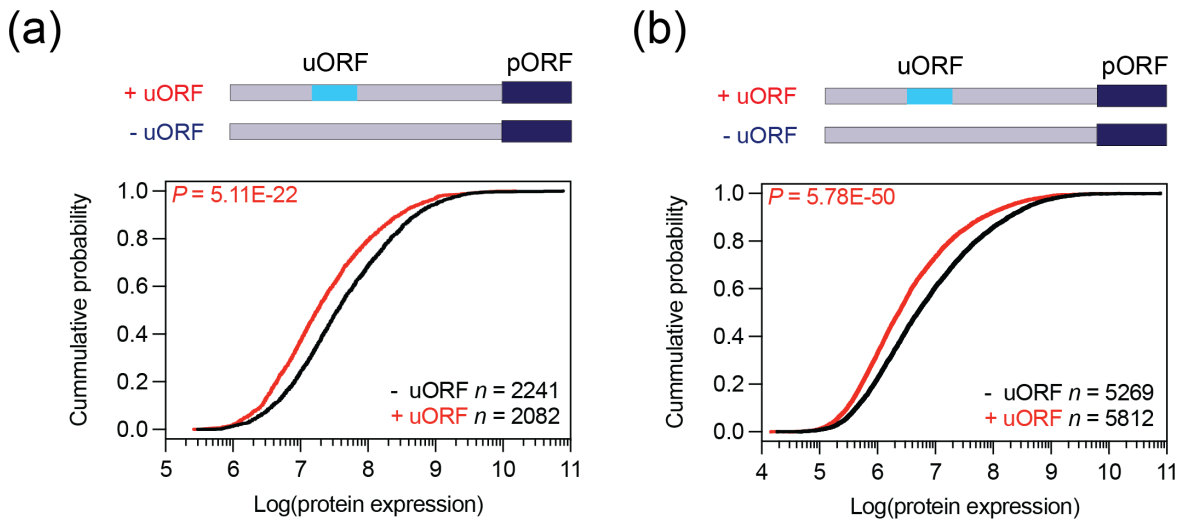


Figure 15. CDF plots of the protein expression of genes that contain or do not contain a uORF

A computational analysis of proteomics data showing the cumulative distribution function of protein expression of genes which contain one or more uORFs (red) and genes which do not contain uORFs (black) (a) Ubiquitously expressed proteins across 29 tissues the contain uORFs have a significantly lower expression (b) In the complete dataset uORF-containing genes have a reduced protein expression in comparison to genes which do not contain uORFs. *P*-values were calculated using the Kolmogorov-Smirnov nonparametric test.

Next, different features were used to classify genes into separate sets, and cumulative distribution function (CDF) plot analysis was performed on the corresponding protein expression data for each of those sets. Due to the complexity of 5′ UTRs with multiple uORFs, some of the feature analysis had to be performed on the transcript population which only has a single uORF in the 5′ UTR.

It was observed that genes with more than 1 uORF tend to have lower protein expression levels (D=0.20, P=2.03e-62) than genes with 1 uORF (D=0.10, P=1.21e-12) **(Figure 16a**), suggesting that genes with multiple uORFs are under stronger repressive regulation by uORFs. Interestingly, uORFs that were out of frame with the pORF were found to be more inhibitory (Frames 1,2 D=0.11 vs. Frame 0 D=0.07) (**Figure 16b)**. One possible explanation for this is that some in-frame uORFs may bypass their termination codon and possibly generate a terminus-extended pORF. Next, minimal uORFs, represented by an AUG initiation codon immediately followed by a termination codon were found to be more repressive (D=0.21, P=0.0056) than uORFs that code for micropeptides (D=0.09, P=3.56e-11) (**Figure 16c)**. uORFs using the UGA stop codon were the most prevalent, and significantly more repressed than uORFs that use UAG or UAA termination codons (D=0.13, P=9.07e-15) (**Figure 16d**).

There was no significant difference between weak and strong Kozak context on transcript expression, suggesting that the Kozak sequence is not a major factor in determining uORF activity; in contrast with previous studies. (**Figure 16e**)[162]. Previous analyses proposing that Kozak may be an important factor show marginal differences;(i.e. $P = 0.04$, (strong context $N = 92$, weak context $N = 439$) has been reported[162]. Finally, uORFs spanning the pORF had similar expression distribution as the population that didn't span the pORF; consistent with previous observations [162]. This result is surprising, as any initiation at a translation initiation site (TIS) spanning uORF eliminates the possibility of reinitiation at the pORF (**Figure 16f**).

A prominent uORF positional effect was observed when uORFs were categorized based on the location of their initiation codons in the UTR. uORFs that appear closer to the cap structure are likely to be less inhibitory than uORFs that initiate closer to the pORF, with uORFs initiating in the last 20% of the sequence space being more prominently repressive (D=0.24, P=1.82e-12) (**Figure 16g**). In fact, the majority of the uORF repressive signal comes from uORFs in the 40% of sequence space closest to the TIS (D=0.18, P=9.66e-24) (**Figure 16h**). Finally, multiple attempts were made to divide uORFs based on their length, in order to generate two sets of similar size. Genes containing uORFs shorter than 51 nucleotides were found to have a lower protein expression in comparison to genes with longer uORFs (D=0.09, P=0.004). Moreover, genes with shorter uORFs were found to have a lower protein expression in comparison to the set of genes which do not contain uORFs (D=0.14, P=7.30e-15) (**Figure 16i**).

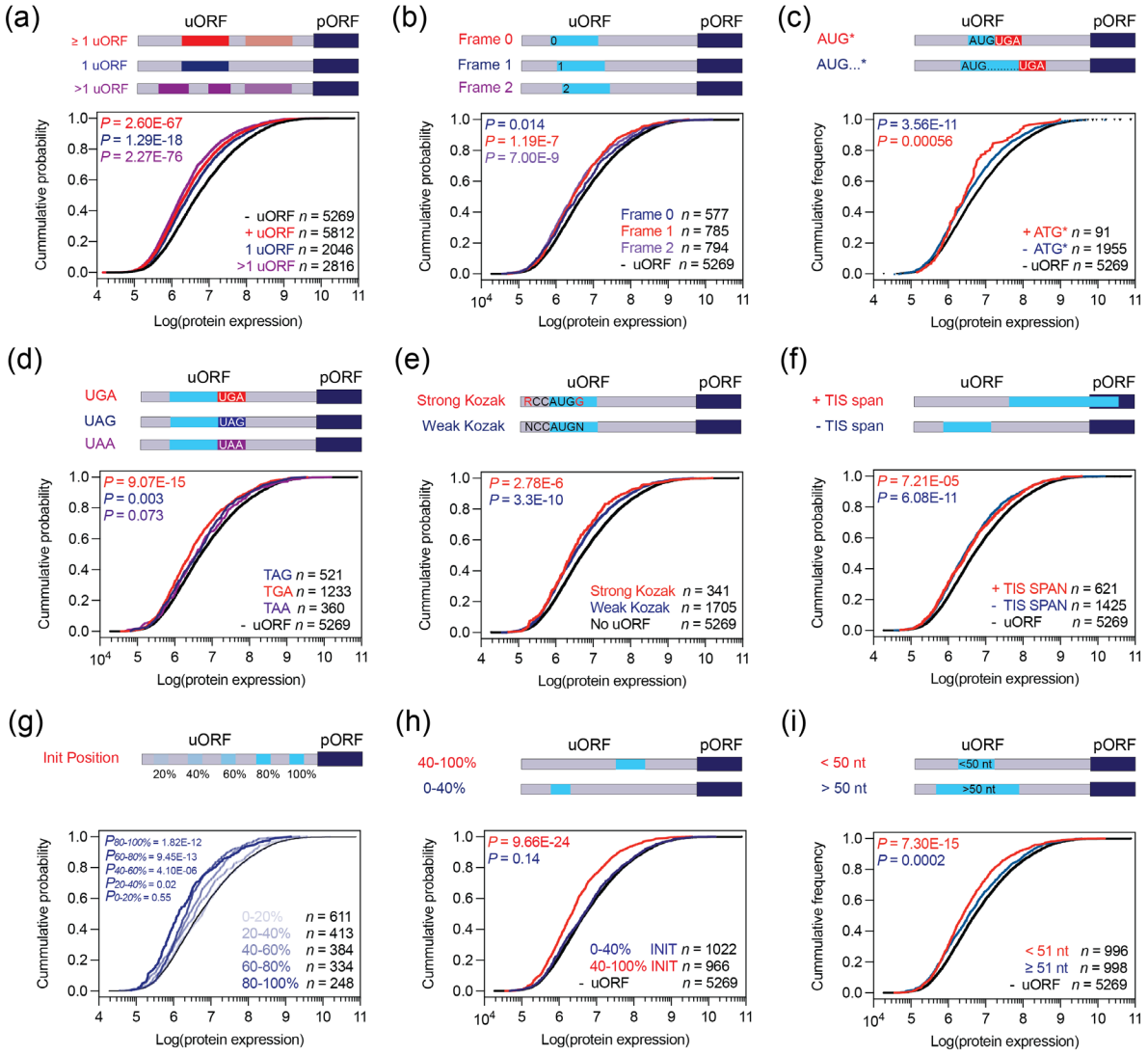


Figure 16. Cumulative probability plots of uORF properties with *P*-values computed using the Kolmogorov-Smirnov test.

Computational analysis of protein expression data on genes categorized based on uORF properties. The CDF of each group was plotted and the KS test was used to calculate a *P*-value and evaluate if its distribution is different from the distribution of the group containing no uORFs (- uORF). Effects of (a) having one or more uORFs in the 5′ UTR (b) uORF frame (c) minimal uORF (d) termination codon (e) initiation context strength (f) uORF sequence spanning pORF (g) uORF start position in 5 bins (h) uORF start position in 2 bins, and (i) uORF length were evaluated.

3.4.4 uORF detective – a novel tool to power the study of uORFs

3.4.4.1 A tool for mapping Ribo-Seq data to the genome (GWIPS)

Ribosome profiling (Ribo-Seq) is a technique that gives a global snapshot of the ribosome locations on the entire transcriptome by pairing the classical molecular method of ribosome footprinting with next generation sequencing[5,325]. In the standard Ribo-Seq footprinting protocol, mRNA is treated with nucleases that degrade every region, but the ribosome-protected discrete footprints which are ~30 nucleotides long[326,327]. The sequenced and mapped footprints give a comprehensive snapshot of *in vivo* translation and are particularly useful for studying uORF activity. Similar protocols based on the ribosome profiling technique allow for a single nucleotide resolution mapping of the TIS[62]. Recently, Kiniry and Michel *et al.* described a tool called Genome-wide information on protein synthesis (GWIPS) that aggregates and visualizes published ribosome sequencing studies[328,329]. The aggregation strategy used in GWIPS enables signal gain, and visualization of mapped reads in areas with low coverage.

GWIPS is a redeployed instance of the UCSC genome browser which hosts a database of ribosome sequencing datasets. Thus, GWIPS provides a platform to map bigwig format Ribo-Seq data to the genome, but not the transcriptome. As a consequence of this, GWIPS is complicated by both direction – since genes are also present on the negative strand, and the presence of introns – which complicates the visualization of exon-spanning features (**Figure 17a**). Additional complexity is introduced by splicing, which introduces large intron gaps in the genome browser view (**Figure 17b**). uORFs which may have the same genomic coordinates, can map differently to isoforms due to splicing. Most critically, GWIPS is not built with the purpose of studying uORFs and is constrained by the functionality provided by the UCSC genome browser instance.

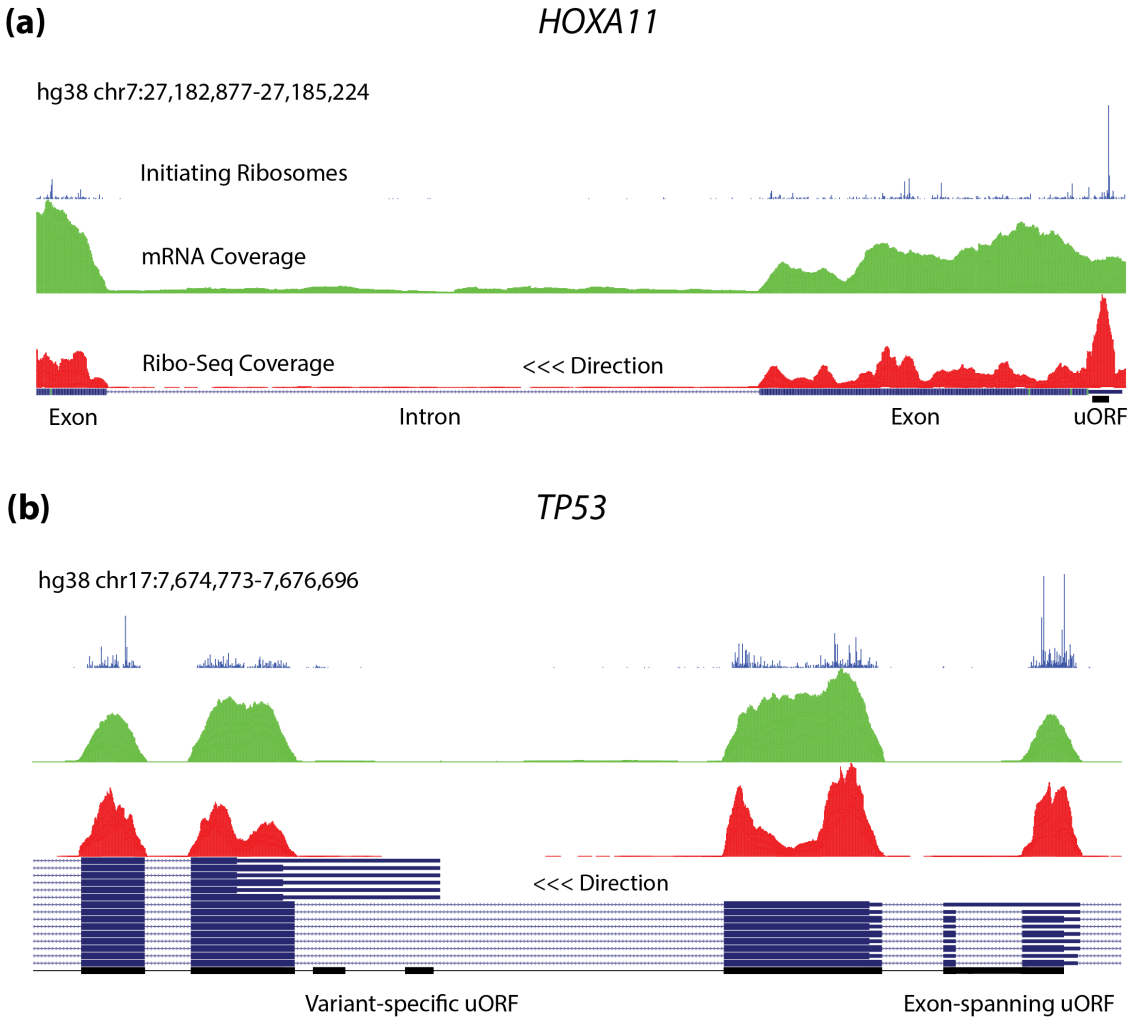


Figure 17. Ribosome profiling and uORF data for the *HOXA11* and *TP53* genes visualized in GWIPS.

Global aggregates of ribosome initiation (Blue), mRNA coverage (Green) and Ribo-Seq footprints (Red) mapped on the human genome (hg38) using the GWIPS software. Both the HOXA11 and TP53 genes are on the minus strand, complicating sequence view. (a) *HOXA11* has a single uORF (black) (b) A selected region of the spliced T53 5′ UTR highlights the complexity of visualizing its uORFs (black) in context of transcript isoforms. Exon-spanning and variant-specific uORFs are shown in context of all transcript variants.

Given that Ribo-Seq is an mRNA sequencing technique, and uORFs are mRNA *cis*-acting elements, a tool that is capable of visualizing Ribo-Seq data, and uORF information on

a transcript level would be of great benefit to future uORF studies. This assessment motivated the development of a custom, interactive, web-based software tool called: uORF Detective.

3.4.4.2 Development and features of the uORF detective software

uORF Detective is a transcriptome-wide bioinformatics tool that combines UCSC and NCBI transcript level features with information from the uORFeome, with large-scale public Ribo-Seq datasets including: 46 Ribosome Footprinting, 8 Ribosome initiation, 34 public mRNA, 4 custom mRNA, and 2 eIF3b bound 40S datasets; all referenced in Baranov *et al* [328,329]. The Ribo-Seq data is presented in tracks, which represent a global aggregate of all related datasets per genomic position. To achieve this efficiently, custom algorithms were built to pre-calculate the global aggregates, as initial versions of the software suffered from delays caused by real-time aggregation calculations. These algorithms utilize the pyBigWig library to output a global bigwig file for each track of interest (footprints, mRNA, initiation, 40S, custom), which represents the sum of the signals from each class of datasets at every genomic coordinate (**Figure 18**)[330].

Several approaches were developed to get the most up-to-date transcript information, including using real-time SQL queries to the UCSC database at runtime. The most recent version can function offline and uses the precalculated uORFeome database as input. Algorithms were developed to efficiently recalculate the uORFeome database using the most up-to-date transcript information of the UCSC genome browser.

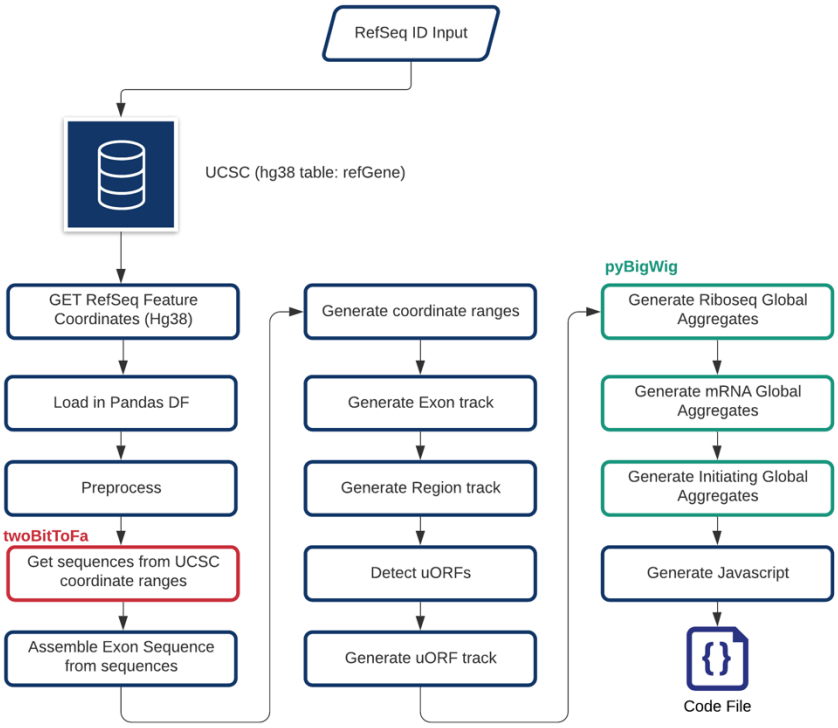


Figure 18. Algorithm process diagram for uORF Detective. The software generates an interactive HTML and a JavaScript data dependency file.

A process diagram of the uORF Detective software, which utilizes the UCSC hg38 genome database to extract transcript features. Generates isoform level sequence information using twoBitToFa,and annotates all uORFs. Global aggregates of ribosome profiling data are generated for initiation, footprints, mRNA coverage and custom data of interest. The software outputs an HTML and a JavaScript dependency file for each splice variant, effectively mapping uORFs and ribosome profiling data on a transcript level.

With this information we can detect all uORFs and generate an output file that can be parsed by the interactive library for visualizing sequence annotations: "The Feature Viewer" which enables the visualization of our processed data in a web browser[331]. By using this tool, the Ribo-Seq data of any transcript of interest can be examined in a 5′ to 3' direction in context of uORFs. For instance, we can see that *HOXA11* has a uORF with significant ribosome density and a strong initiation peak. In fact, uORF initiation appears to be preferred to that of the pORF site at this locus (**Figure 19**). uORF Detective features multiple elements including:

Gene and transcript IDs with versions (**Figure 19-1**), Transcript annotations for UTRs, CDS, and Exon map (**Figure 19-2**), a uORF track with hover display of: length, frame, termination codon use (**Figure 19-3**), two near-cognate initiating uORF tracks for (CUG and GUG), only if the uORFs are under strong Kozak consensus (**Figure 19-4**), Ribosome footprint track (**Figure 19-5**), Initiating ribosomes track (**Figure 19-6**), 40S subunits track (**Figure 19-7**), and an mRNA track (**Figure 19-8**). In addition to this, uORF Detective calculates and visualizes the relative ratio of uORF initiation relative to the pORF initiation and normalized by the mRNA track. The reason for this is to account for the fact that some parts of the UTR have low sequencing coverage. While this metric is not ideal, it allows for a rapid detection of initiating uORFs computationally on a transcriptome scale. uORF detective is a fully interactive web-based tool, which can drive future uORF research efforts and enable rapid evaluation of transcripts in context of their uORFs and ribosome profiling data.

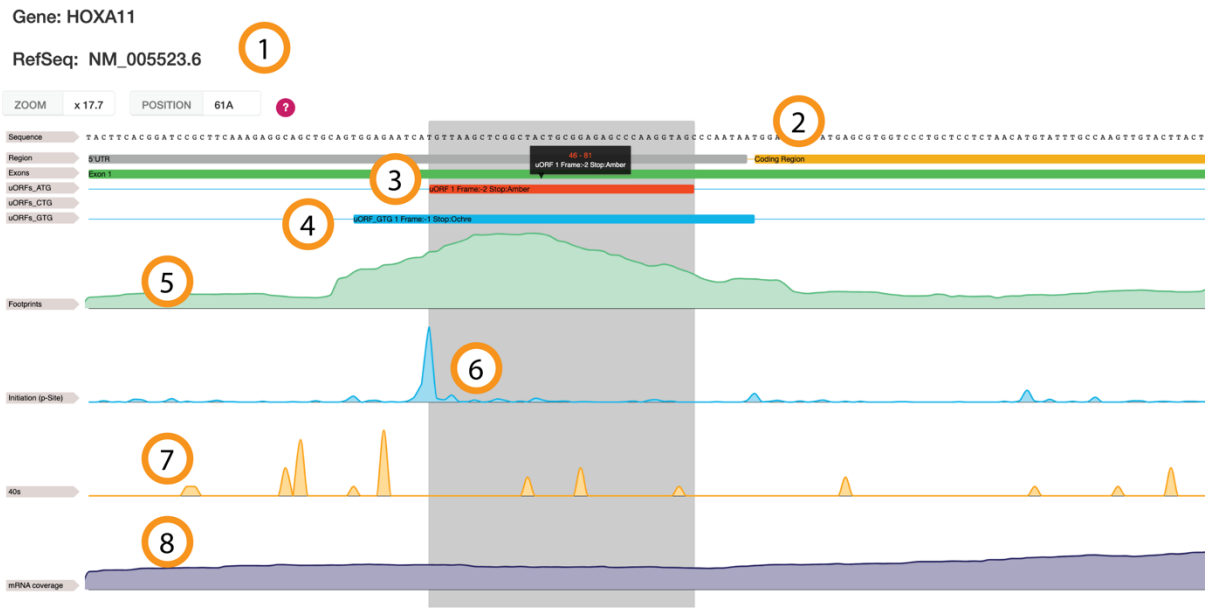


Figure 19. Ribosome profiling data of HOXA11 visualized in uORF Detective.

Computational analysis of uORFs in context of ribosome profiling data using uORF Detective. The *HOXA11* variant NM_005523.6 is used to demonstrate the following uORF Detective features: (1) Gene and transcript IDs with versions (2) Transcript annotations for UTRs, CDS, and Exon map (3) uORF detection, with length, frame, termination codon use (4) CUG and GUG uORFs under strong Kozak consensus are mapped (5) Ribosome footprint track (6) Initiating ribosomes track (7) 40S subunits track (8) mRNA track.

3.4.4.3 ATF4 and uORF-mediated differential regulation of translation during stress conditions

It was previously discussed that the availability of Met-tRNA$_i^{Met}$•eIF2•GTP ternary complex (TC) is critical for translation initiation, and that under normal conditions eIF2 is recharged with GTP by eIF2B. However, when the integrated stress response (ISR) is activated, a serine 51 phosphorylation of the eIF2α subunit, turns eIF2 into a competitive inhibitor of eIF2B, which prevents eIF2 recycling. As a result of eIF2α phosphorylation the levels of available TC drop, which ultimately leads to repression of global translation[26,55].

Several uORF-containing genes are preferentially translated during ISR activation including: *ATF4*, *CHOP*, *ATF5*, *C/EBP*α and *C/EBP*β ; all transcription factors that activate gene expression cascades involved in addressing cellular stress[212]. These findings suggest that uORFs – which have been generally associated with translational repression of the pORF, may also serve as pORF translation activators under specific conditions.

The observation that uORFs can differentially regulate translation under stress conditions has been best exemplified by research on the yeast *GCN4* and human *ATF4* genes; two of the best characterized genes in context of uORF research[169,171,174,332]. These studies clearly link the ability to activate pORF translation during ISR with the presence of particular uORF configurations, which are normally repressive when TC levels are abundant.

Namely, the *ATF4* gene is thought to contain two uORFs, a 3-codon uORF 1 with a stop codon in the 5′ UTR, and a 59-codon uORF 2 which has its initiation codon in the 5′ UTR but ends in the CDS, spanning the pORF initiation codon (**Figure 20a**). Conceptually, it can be expected that initiation at uORF 2 will result in unproductive ribosomes in context of pORF translation, as these ribosomes have bypassed the pORF initiation codon.

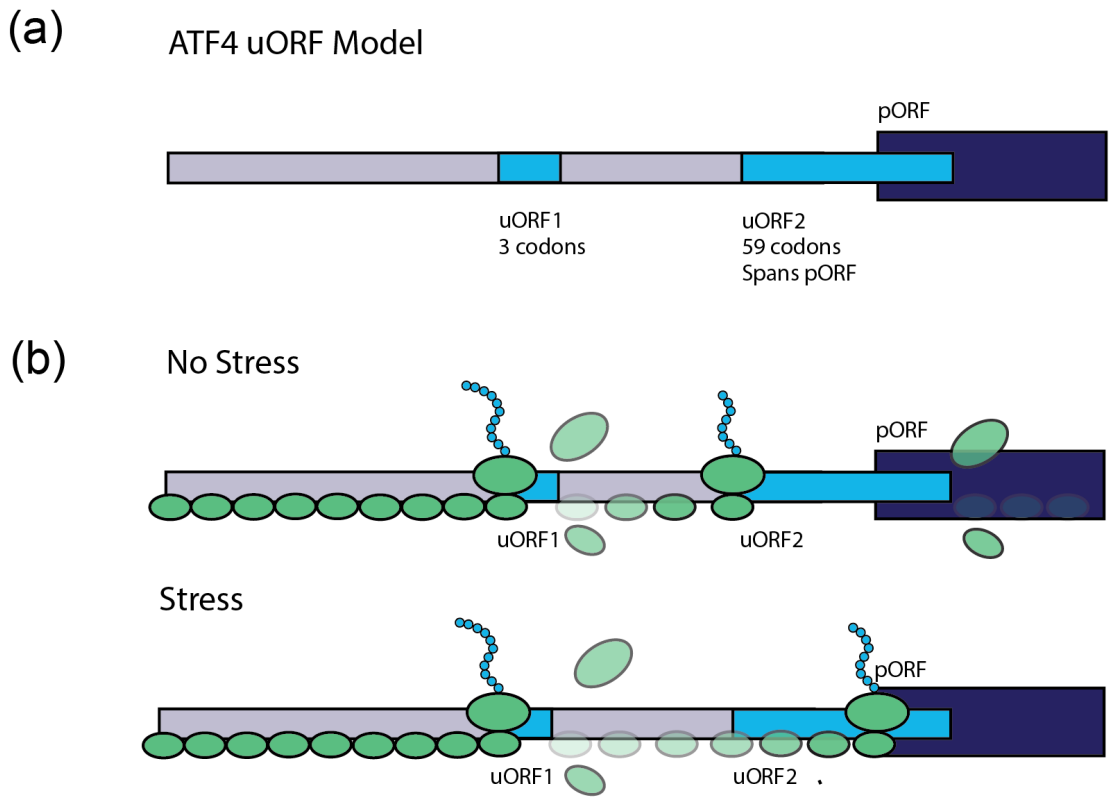


Figure 20. Published model of *ATF4* translation during normal and stress conditions.

A schematic representation of the *ATF4* gene 5′ UTR and its established translational model under normal and stress conditions. *ATF4* contains two uORFs, where uORF 1 is fully enclosed in the 5′ UTR, while uORF 2 spans the pORF initiation codon (b) Under normal conditions the scanning ribosome initiates at uORF1, 40S ribosomal subunits regain competency before initiating at uORF 2 resulting in repression. Under stress conditions 40S subunits take longer time to recharge with TC and bypass the uORF 2 initiation codon, initiating at the pORF.

Under normal conditions, the *ATF4* gene is thought to initiate at its first uORF, at which point the 40S ribosome undergo either complete, or incomplete ribosome recycling. The 40S ribosomal subunits which remain engaged with the mRNA continue to scan for a subsequent codon, and get recharged with a Met-tRNA$_i^{Met}$•eIF2•GTP ternary complex, in time to initiate at uORF 2. Successful initiation at uORF 2 has an inhibitory outcome for translation at the pORF, as ribosomes which initiate at uORF 2 are likely to bypass the pORF initiation codon (**Figure 20B**)

Recently it was found that stress induced ATF4 translation is dependent on eIF2d and DENR, revealing that reinitiation is a critical aspect of ATF4 translation. Namely, it was observed that when HAP1 cells where *eIF2D* was deleted using CRISPR-Cas9 editing translation, were treated with the ER stress-causing chemical Tunicamycin, translation of ATF4 was reduced. Similar results were observed when DENR was knocked down, suggesting that both eIF2d and DENR are essential for stress induced ATF4 translation[170]. The function of eIF2D and DENR is to destabilize the codon-anticodon pairing at the P-site, resulting in the release of the tRNA and mRNA[121,123]. However, these results suggest that the 40S ribosome must stay engaged with the mRNA. Interestingly, it has been reported that eIF2D, DENR and MCTS-1 can potentially serve as noncanonical carriers of Met-tRNA$_i^{Met}$, which suggest that they may play a role in recharging 40S ribosomal subunits[195,333].

Given that *ATF4* is one of the best described examples in context of uORF regulation of translation, a uORF Detective analysis was performed to assess its translational profile. This analysis revealed a surprising result – showing that most of the initiating ribosomes accumulate at a previously undiscovered minimal uORF at position 25 (**Figure 21**). Ribosome footprints are also significantly enriched in the ~30 nt sequence covering this uORF. The importance of minimal uORFs may have been underestimated; likely due to lack of elongation, as minimal uORF are defined as an initiation codon, which is immediately followed by a termination codon. For the purpose of comparison with previous analyses of *ATF4*, the minimal uORF will be referred to as: uORF 0. A new model that includes this minimal uORF was proposed (**Figure 22**).

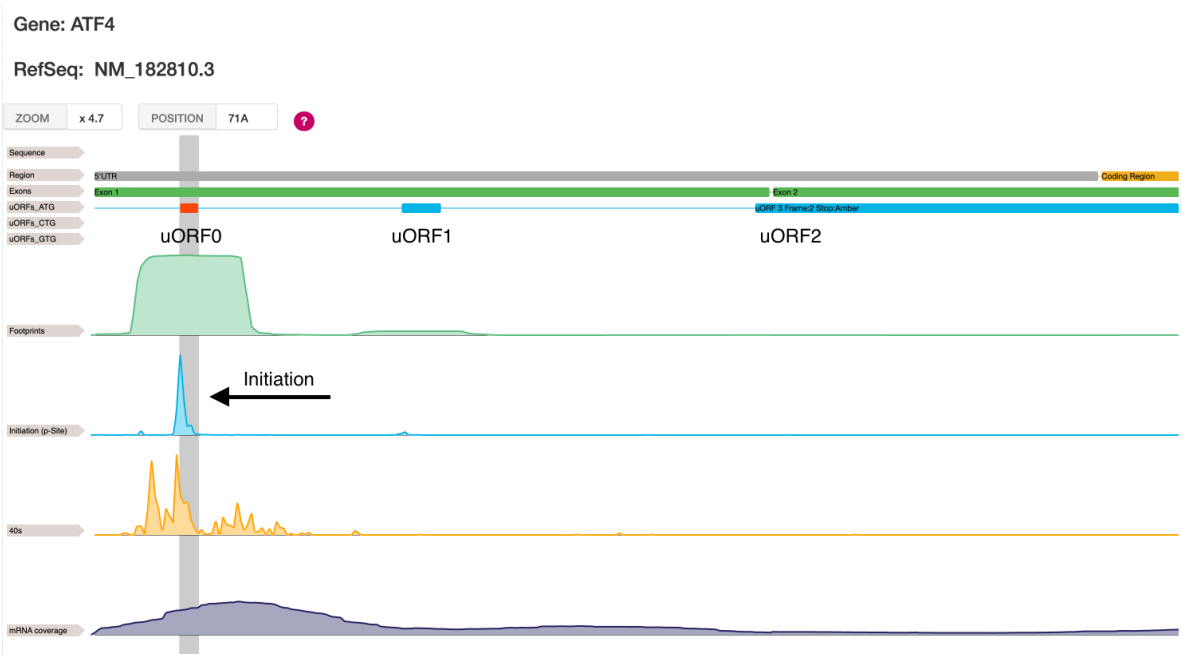


Figure 21. uORF detective analysis of the *ATF4* gene

A uORF Detective analysis of the *ATF4* gene showing that initiating ribosomes are mapped to a novel minimal uORF 0 (Initiation - Blue). The ribosome footprints also aggregate at this location (Footprints - Green). Given this observation, a new model can be proposed, where *ATF4* is regulated by 3 uORFs in the 5′ UTR.

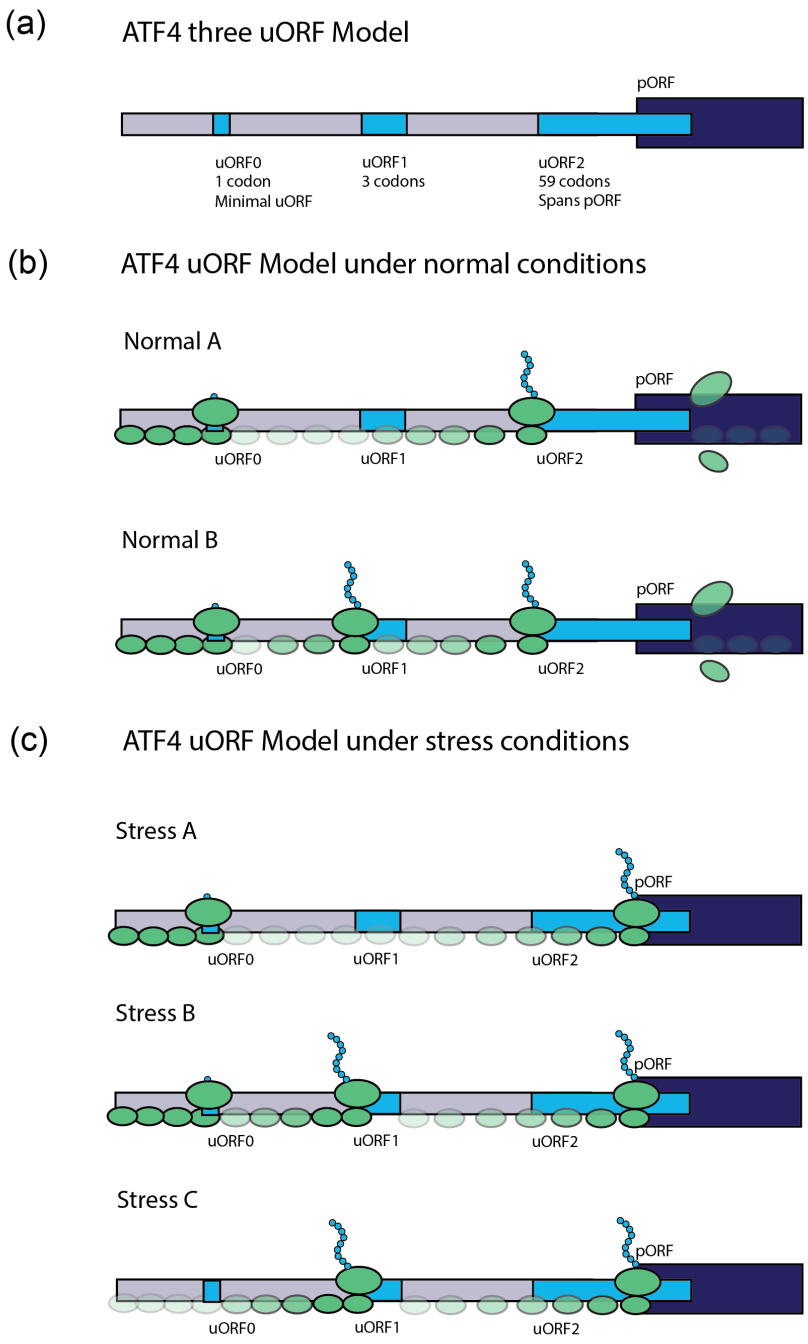


Figure 22. A proposed translation model for the *ATF4* gene.

A schematic representation of the *ATF4* translational model. *ATF4* is regulated by 3 uORFs, a one codon minimal uORF 0, a three codon uORF 1 and a 59 codon long pORF spanning uORF 2. (b) Under normal conditions the ribosome likely initiates at uORF 0 followed by reinitiation at uORF 2 or reinitiation at uORFs 1 and 2. (c) Under stress conditions (Stress A) 40S subunits may initiate at uORF 0 and bypass uORFs 1 and 2, initiating at the pORF, initiate at uORF 0, uORF 1 and then bypass uORFs

2 to reinitiate at the pORF, or uORFs may bypass uORF 0, initiate at uORF 1, and reinitiate at the pORF.

3.4.4.4 Validation of a new translational model for the *ATF4* gene

Given that the *ATF4* gene is one of the best described genes in context of uORF regulation, it was prudent to validate these findings. For this purpose, a strategy using a combination of synthetic gene blocks and site-directed mutagenesis (SDM) was used to clone the *bona fide* 5′ UTR of *ATF4*, as well as hypothetical analogs of the *ATF4* 5′ UTR where the uORF configuration was altered by mutating the uORF initiation codons, into the pDLR 5′ UTR vector. uORFs were inactivated by converting their initiation codons from ATG to TTG (on a DNA level), resulting in "silencing" of uORF recognition due to a change to a less favorable near-cognate initiation codon. Studies in HEK293T cells have shown that initiation codons are used with variable efficiency in comparison to the canonical AUG initiation codon. The CUG codon is the second most active initiation codon with (19%), followed by GUG (9%) and UUG (2%)[82,334]. Thus, it can be expected that for the modification of uORF initiation codons from AUG to UUG to result in a near complete silencing of initiation; effectively allowing the 40S ribosome to scan through the uORF initiation codon.

To test the effects of individual uORFs on the translation of the pORF of the ATF4 gene, we generated plasmids for all of the possible uORF combinations, where uORFs initiation codons were left as wild-type/ATG (white circle), or mutant/TTG (black circle) (**Figure 23**).

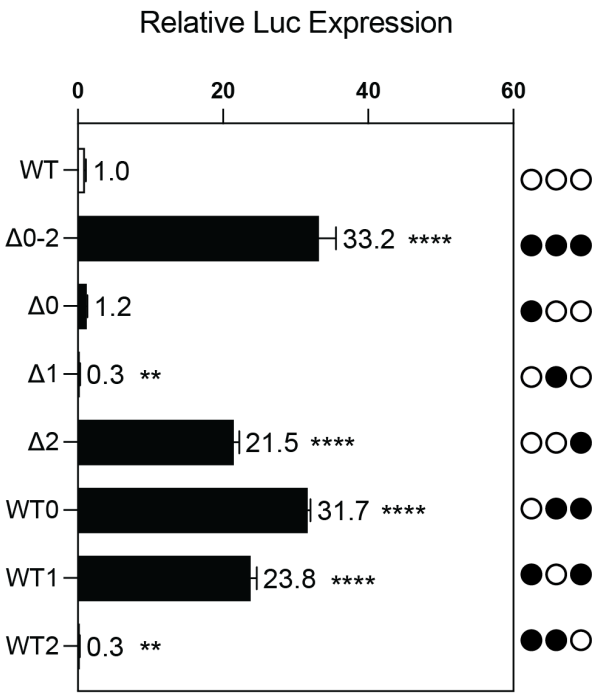


Figure 23. Modeling the uORF regulation of the *ATF4* gene

The three uORFs of the *ATF4* gene were mutated in different configurations, where white circle represents a wild-type AUG codon, and black circles represent silenced uORFs with a UUG initiation codon. Error bars represent ±SD of ≥4 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (**$P \leq 0.01$, ****$P \leq 0.0001$).

First, the uORFs in *ATF4* significantly repress translation of the pORF (WT), as mutating the initiation codons of all three uORFs (Δ0-2) results in a 33.2-fold increase of RLuc production. This result highlights the significant potential for protein upregulation by interfering with uORF initiation.

Next, when the minimal uORF 0 is silenced (Δ0), there is only a slight increase in efficiency at the pORF, suggesting that in this case the ribosomes initiate efficiently at uORF 1 and then uORF 2, with a small number of ribosomes that fail to reinitiate at uORF 2 likely initiating at the pORF. Interestingly, when the second uORF 1 (Δ1) is silenced, there is an

even stronger repression of the pORF. This suggests that under normal conditions uORF 1 promotes some initiation at the pORF through bypass of uORF 2. As expected, the silencing of uORF 2(Δ2) has a significant positive impact on translation at the pORF, as uORF 2 spans the pORF initiation codon. In fact, in all cases when uORF 2 is silenced (Δ0-2, Δ2, WT0, and WT1), a significant increase in signal coming from the RLuc pORF was observed, confirming the known inhibitory effects of uORF 2[170,171]. The striking similarity between the pairs Δ0-2 and WT0, Δ1 and WT2, Δ1 and WT1 may suggest that uORF 0 is not active. However, despite the lack of evidence for significant influence of uORF 0, a model is required to reconcile the strong initiation signal observed in the Ribo-Seq analysis. Under this new hypothetical model, uORF 0 may promote both a strong initiation, and efficient reinitiation (**Figure 24**). Given that the bioinformatics analysis on the proteomics data suggests that minimal uORFs are as inhibitory as translated uORFs, more work is required to understand the reason for inclusion of uORF 0 and the potential role it may play under different cell conditions.

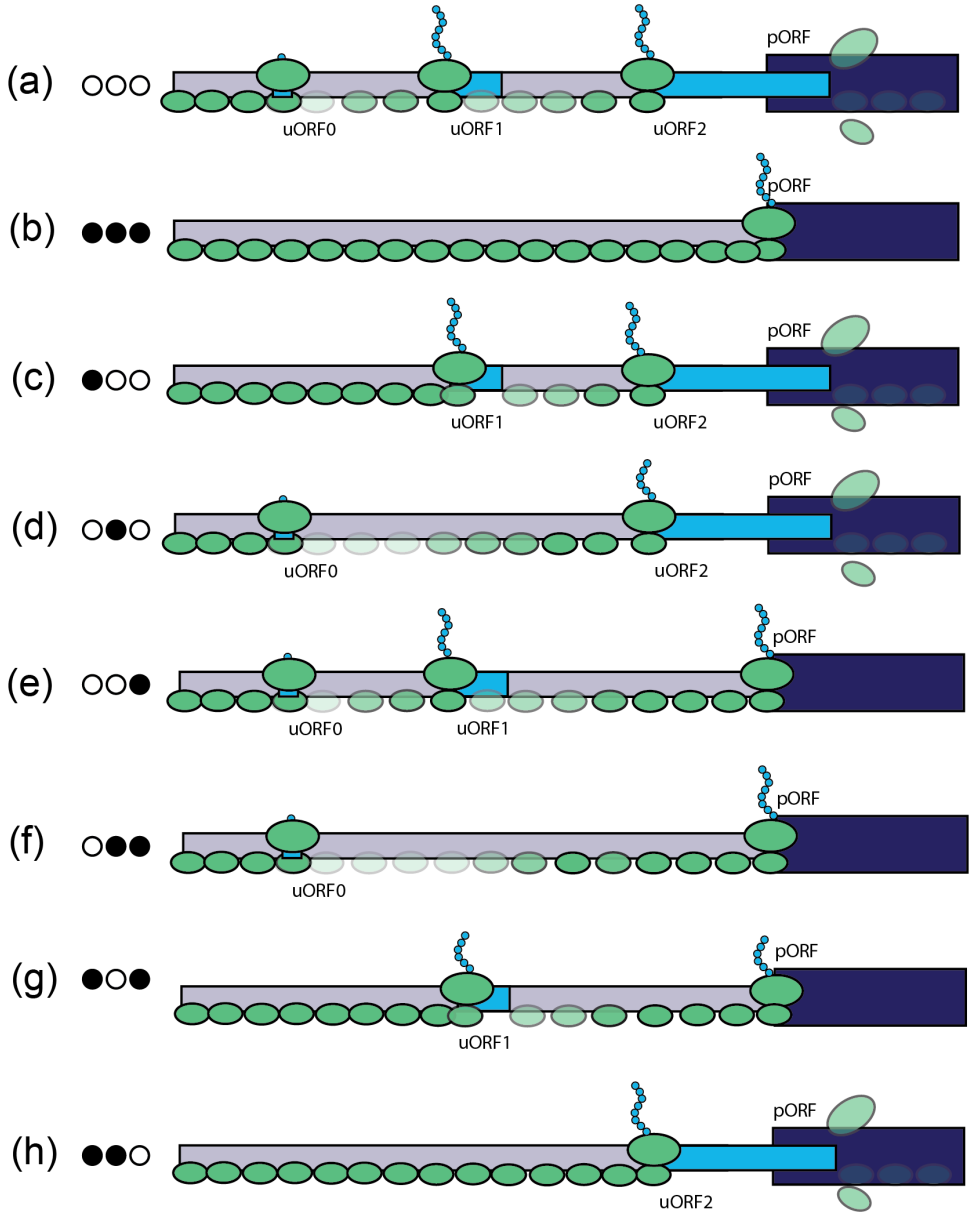


Figure 24. A proposed model for *ATF4* translational regulation

uORFs initiate at uORF 0 and can reinitiate at uORF 1 and 2 leading to repression at pORF (b) Optimal 5′UTR has no inhibiting uORFs, results in pORF translation (c) Removal of uORF 0 leads to initiation at uORF 1 and reinitiation at uORF 2 which leads to repression (d) Removal of uORF 1 results in uORF 0 to uORF 2 reinitiation, resulting in repression at pORF (e) Removal of uORF 2 leads to translation of uORFs 0 and 1, and an initiation at the pORF (f) Removal of uORFs 1 and 2 leads to translation of uORF 1 and reinitiation at the pORF (g) Removal of uORFs 0 and 3 leads to initiation at uORF 1 and reinitiation at the pORF (h) removal of uORFs 0 and 1 results in initiation at uORF 2 which results in repression of the pORF.

3.4.5 Predicting the repressive potential of uORFs

Luciferase-based plasmid systems are a powerful tool to model the effects of *cis*-acting elements. This approach can reveal valuable information on uORF activity and their repressive potential. However, like all *in vitro* experiments, cloning strategies require significant time and resources, and are therefore limited in the number of hypotheses that can be tested. Artificial intelligence may provide an alternative way to test hypotheses in the field of biology. For instance, the ability to reliably predict the regulatory potential of *cis*-elements sequences *in silico*, could lead to interesting discoveries which can subsequently be validated with small-scale *in vitro* experiments.

Recently a massively parallel reporter assay (MPRA) was used to measure the impact of randomly generated 5′ UTR sequences on translation[213]. The strategy employed the generation of a library of 280,000 sequences fused to an enhanced green fluorescent protein sequence (eGFP). The polysome profiling technique was used to measure mean ribosome load (MRL), a metric estimating the average number of ribosomes loaded on each mRNA[213]. This data were then used to train a convolutional neural network deep learning model referred to as Optimus 5-prime, and capable of predicting MRL when given a sequence as input. The flexible length version of the model was trained on random sequences 25-100 nt in length, and is therefore limited to testing of sequences that fit in this range.

Conceptually, the Optimus 5-prime model has been exposed to ~260,000 sequences each associated with a MRL value, and through the training process has generated a network of sequence features, relationships, and rules which may not be immediately obvious or easily characterized by targeted bioinformatics analyses[213]. These features include, but are not limited to initiation and stop codons, uORFs, and GC-rich regions, likely involved in the formation of secondary structures.

Despite limitations with the training data length, this model may be used for testing hypotheses of synthetic sequences.

3.4.5.1 Predicting the regulatory effects of 5′ UTRs from sequence

Given that the Optimus 5-Prime model has learned about sequence features such as initiation codons, stop codons, secondary structures and uORFs, it was initially deployed to test predictions against previously generated *in vitro* experimental data. The pDLR GOI 5′ UTR vector used in these experiments was previously described in context of cloning the *ATF4* gene. This vector was used to clone the *HOXA11* WT (containing an active uORF) and separately the *HOXA11* TTG (containing a mutated initiation codon ATG->TTG). The placement of the *HOXA11* uORF under a weak UUG initiation codon results in significant derepression of translation at the RLuc pORF (**Figure 25c**). Interestingly, the model predicted a significantly higher MRL level for the mutated uORF (MRL = 6), than the wild-type uORF (MRL = 3.7). This prediction suggests that more ribosomes would be loaded if the uORF initiation codon is mutated from AUG to UUG. This demonstrates that the model was able to recapitulate the result of an *in vitro* experiment. It should be noted that the MRL values and the normalized luminescence are different measures, and that while their relationship is expected to be positively correlated, it has not been empirically established[213]. However, both MRL and relative luciferase expression are modeled measures for translation efficiency.

The next set of tested sequences utilizes another vector pDLR *HOXA11* 5′ UTR uORF which uses the pDLR *HOXA11* 5′ UTR WT. This vector is designed to enable the replacement of the HOXA11 uORF sequence and test hypotheses related to the uORF encoded sequence (**Figure 25a**). It was observed that successive glutamic acid repeats (Poly-E) in the sequence encoding the uORF has an inhibitory effect on the translation of the RLuc pORF (**Figure 25c**).

This observation is striking, as it demonstrates that the uORF encoded sequence may be an additional component that may influence translational regulation. However, here it should be noted that while this kind of homopolymers do not occur naturally in the transcriptome, this knowledge may be applicable in tuning of translation of genes in artificial expression vectors. Interestingly, the machine learning model generally predicted an inverse relationship between MRL and the introduction of longer poly-E sequences in the uORF; consistent with experimental observation (**Figure 25b**). Given that the model has an accuracy (a measure of prediction performance) of ~80% some variability is expected. However, it is evident that the model has learned important features concerning the relationship between uORF identity and repressive activity.

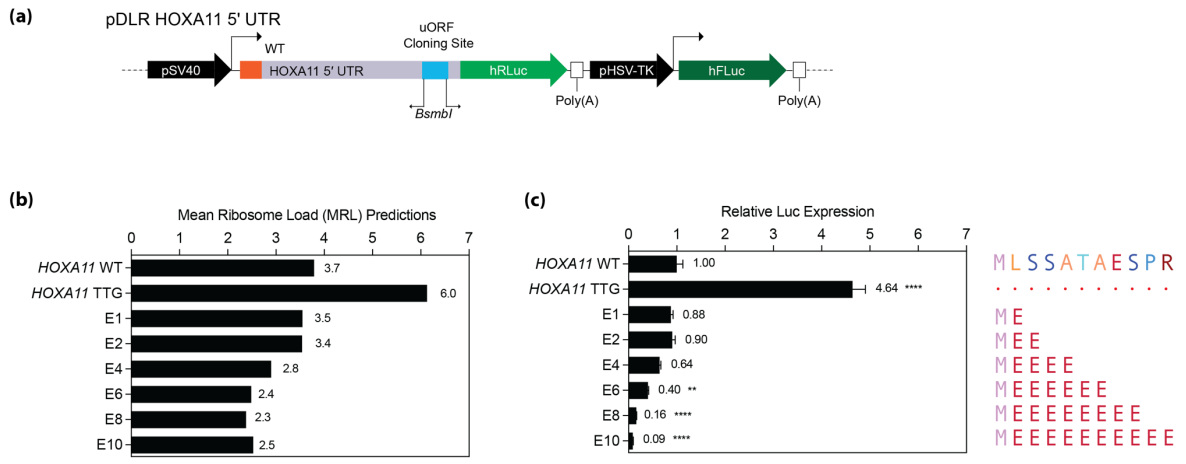


Figure 25. Machine learning predictions of poly-E repression

Machine learning predictions (left) and report assay validation of uORF sequences containing increasing glutamic acid amino acid content. (a) The pDLR *HOXA11* 5′ UTR vector is used to replace the sequence of the *HOXA11* uORF with a uORF sequence of interest. (b) Machine learning mean ribosome load predictions and (c) corresponding reporter assay experimental data generated by Dr. Britt Hanson evaluating successive expansions of glutamic acid effect on RLuc translation.

3.4.6 Probing the Optimus 5-Prime model for novel uORF features

3.4.6.1 Generating MRL predictions for two million randomly generated uORFs

One of the challenges in deep learning is extracting the rules that the model has learned. The visualization of several filters of the Optimus 5-Prime model first and second convolution layers shows that the model has discovered signals related to initiation (AUG codon), termination (UAA), their combination (uORF), and some repeat sequences including poly-U and GC-rich regions[335]. This process is complex, and only reveals motifs the model has found to be critical, without any specific context. A strategy was designed to extract information from the model by asking it to make predictions on a large-scale using artificially simulated data. Namely, given that the model can evaluate the relationship between sequence and MRL, 2 million sequences were computationally generated and their MRL values were predicted. Conceptually, this is a simulation of cloning a random uORF in the pDLR HOXA11 5′ uORF while keeping the 5′ UTR context, as well as the uORF initiation and termination codons unchanged (**Figure 26**). It is important to note that a simulation of 2 million sequences represents a small fraction of all possible sequence combinations which is 4^{30} = 1.1529215e+18; representing 4 possible nucleotide at 30 positions, while keeping the initiation and termination codons unchanged. The selection of 2 million as a sample size was guided by access to computational resources and processing time. Despite this limitation, after five separate random simulations the outcomes were similar, suggesting that the observed signals were meaningful.

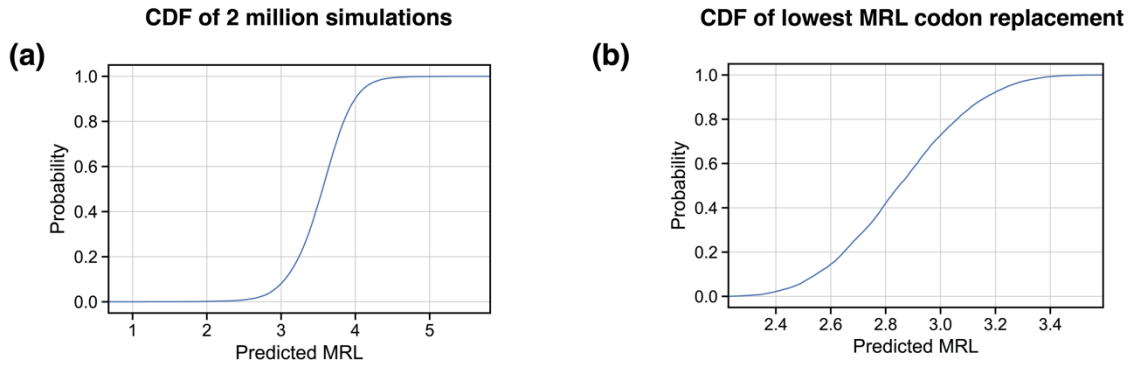


Figure 26. MRL predictions on simulated uORFs in the HOXA11 5′context

Machine learning predictions of MRL for two million randomly generated uORFs, predicted in the *HOXA11* native 5′ UTR sequence context. Two million simulated sequences were used to replace the *HOXA11* uORF (a) CDF plot of the predicted MRL for each sequence shows a normal distribution centered near the *HOXA11* WT predicted MRL = 3.7 (b) A codon replacement strategy applied to the most inhibitory sequence shows increase of MRL rather than further decrease.

The predicted MRLs were processed using custom algorithms that map the amino acid sequence and amino acid properties, and were visualized with the Python Logomaker library[336]. The sampling omitted using Methionine or termination codons in the internal uORF encoded sequence in order to prevent the introduction of additional internal uORFs or premature termination elements. A separate analysis using internal Methionines (M) was also performed with the lowest MRL predicted to be 0.02 for the sequence MASIMDLPGWG. In the most inhibitory internal methionine-containing sequences, a methionine was present near the center and end/or last position of the uORF amino acid (**Figure 27**). This result may be expected as alternative out of frame M can introduce pORF-spanning uORFs.

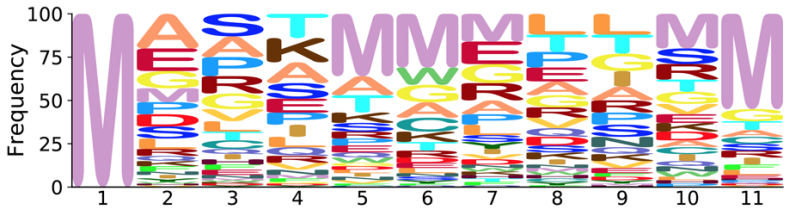


Figure 27. Sequence logo representing the amino acids frequency of the 100 most inhibitory internal Methionine-containing sequences

Machine learning predictions of MRL on two million randomly generated uORFs predicted in the *HOXA11* 5′ UTR context, where internal Methionines were allowed. The top 100 most inhibitory sequences feature internal methionines; most prominently at positions 5, 6, 7, 10, and 11. Due to the position of the *HOXA11* uORF internal methionines may effectively create overlapping uORFs which may span the pORF initiation codon.

When internal Methionines were omitted, the model was able to predict MRL across a broad range, suggesting that some sequence elements are inhibitory, while others are not. The most inhibitory sequence in this analysis was found to be MDRPRWRRRPE, with a predicted MRL of 0.68 (**Figure 26a**).

Next the mRNA sequence of the MDRPRWRRRPE most inhibitory uORF was resampled by randomly replacing codons but keeping the amino acid sequence the same. Codon replacement created another broad distribution but was found to on average result in higher uORF MRL predictions (**Figure 26b**). This may suggest that the initial most inhibitory sequence was already optimized, or that the model in this particular example considers other features to contribute more towards the predicted inhibitory effect than codon usage bias.

3.4.6.2 AUG -> UUG mutation of the initiation codon is predicted to result in increased MRL

Given that the key rate-limiting step of translation is initiation, interfering with uORF initiation by weakening the uORF initiation codon is expected to result in uORF initiation codon bypass and continued scanning towards the next optimal TIS. To test this hypothesis, for every uORF of the two million predictions, a corresponding uORF was created changing the AUG codon to a weaker UUG codon. Custom Kernel Density Estimation (KDE) and low alpha algorithms were developed in the Python programming language to plot each AUG MRL prediction with its corresponding UUG MRL prediction.

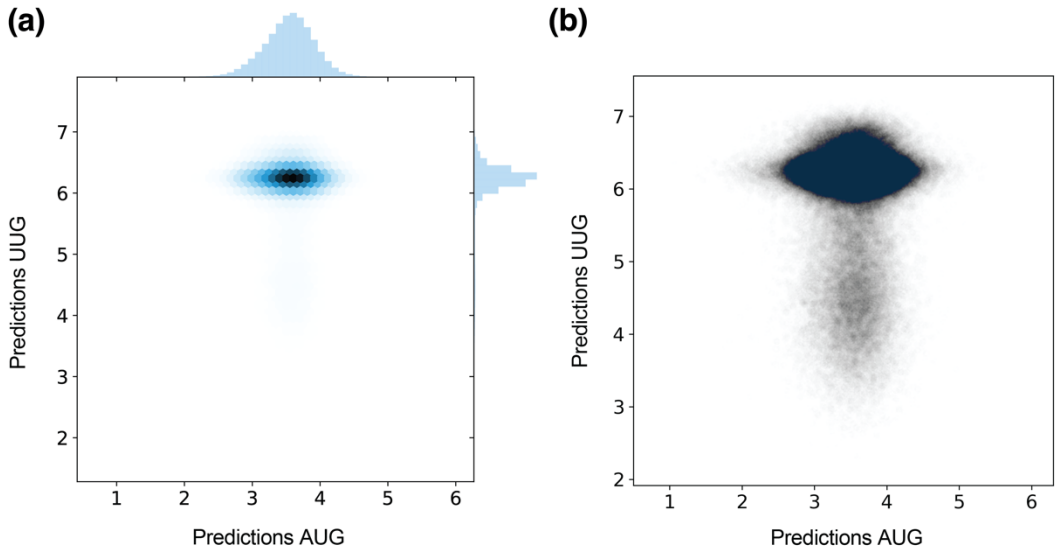


Figure 28. XY plots of the predicted MRL of two million sampled AUG-initiated uORFs and their UUG-initiation mutated analogs

(a) KDE plot shows the center of the data shifts towards higher MRL when AUG is changed to UUG (b) Complete XY plot with reduced transparency shows a minor population that is not sensitive to UUG modification.

These findings suggest that modifying the uORF initiation codon from AUG to UUG, is predicted to result in an increased MRL, which may be a result of uORF initiation codon bypass. Interestingly, while the AUG predictions have a broader MRL distribution, the UUG introduction effect is more constrained. Hence, it can be proposed that while a randomly generated uORF may inhibit downstream translation reinitiation, modifying the AUG initiation of the uORF is expected to reduce the inhibitory effect (**Figure 28a**).

Given that KDE plots estimate the center of the data, an unbiased visualization was generated by plotting every XY coordinate for the AUG/UUG related uORF sequences. The transparency was reduced to try to observe any populations that don't fit the KDE center. Interestingly, it was found that a small population of the data was insensitive to the UUG initiation mutation. This may suggest that some sequence elements in the uORF may enhance UUG initiation (**Figure 28b**)

3.4.6.3 Common feature analysis of the tails of the distribution reveals novel motifs

The randomly sampled uORF library resulted in a normal distribution of predicted MRL values for two million sequences. To extract the most represented features, the positional frequency of the amino acids was analyzed in the top 100 lowest MRL, and the top 100 highest MRL. Custom algorithms were developed to calculate the frequency of the amino acids and evaluate the corresponding amino acid properties. This analysis revealed a prominent Tryptophan (W) signal at position 5 in the most inhibitory sequences (**Figure 29a**). Analysis on the least 100 inhibitory sequences revealed an Arginine-Tyrosine-Glycine (RYG) motif and Aspartic acid (D), or more generally enrichment of charged amino acids in the tail of the uORF sequence (**Figure 29ab**). It is important to note that sequence logos represent positional frequency distribution plots, which means that some of the sequences may only have a partial representation of the overall motif (**Figure 29c**).

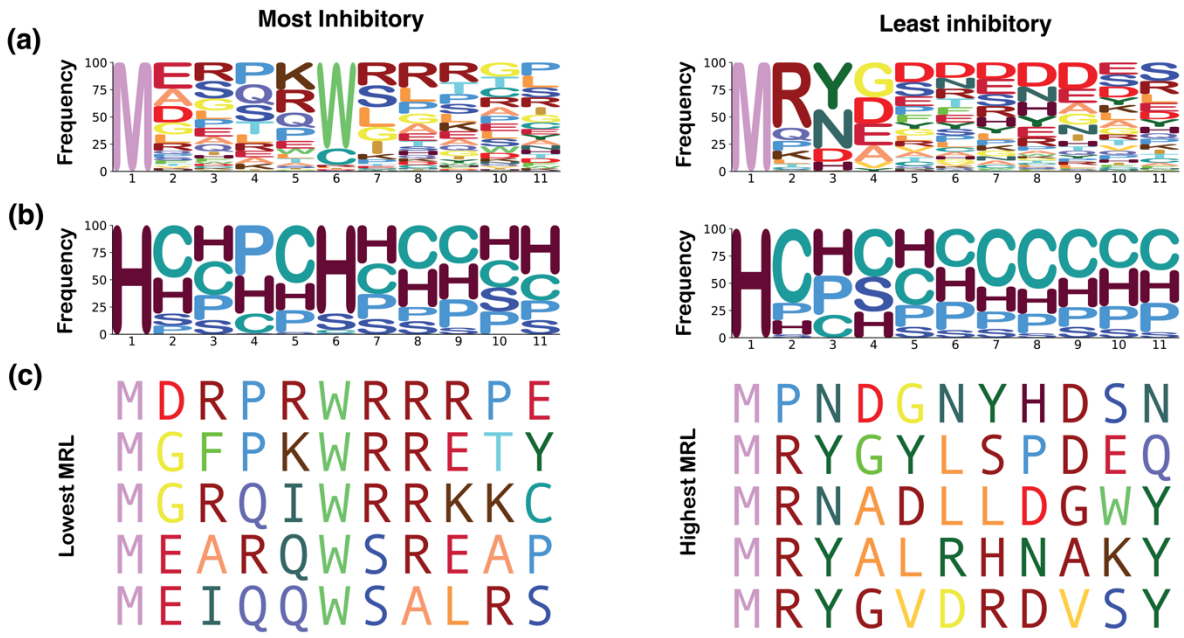


Figure 29. The regulatory potential of uORFs may be influenced by *cis*-elements in the uORF-encoded sequence

Sequence logos of the top and bottom 100 sequences from two million predictions on randomly generated uORF sequences evaluated in the *HOXA11* 5′ UTR context reveal distinct motif signals. (a) A Tryptophan (W) at position 5 is a common feature of the most-inhibitory, lowest MRL group, while an Arginine-Tyrosine-Glycine (RYG) motif and Aspartic acid (D) enrichment were observed in the least inhibitory – high MRL subset (b) Amino acid sequence logos describe (Hydrophobic, Polar, Charged, and Special) frequencies. (c) The top 5 and bottom 5 uORF amino acids show that the complete motifs are not necessarily present in all sequences.

3.4.6.4 Investigating amino acid influence in the most inhibitory sequence

The contribution of each amino acid was investigated by an *in silico* Alanine scan, where each amino acid was replaced with Alanine and the MRL of the sequence was predicted. The MRL of the original sequence (MDRPRWRRRPE) was found to be 0.68. When individual amino acids in this sequence were replaced with Alanine, the MRL predictions of most modified sequences were not significantly affected. However, changes at positions 4, 5, and 6, and in particular the removal of the Tryptophan (W) at position 5 and Arginine (R) at position

6 resulted in a 4.3-fold increase in predicted MRL in comparison to the unmodified sequences (**Figure 30**).

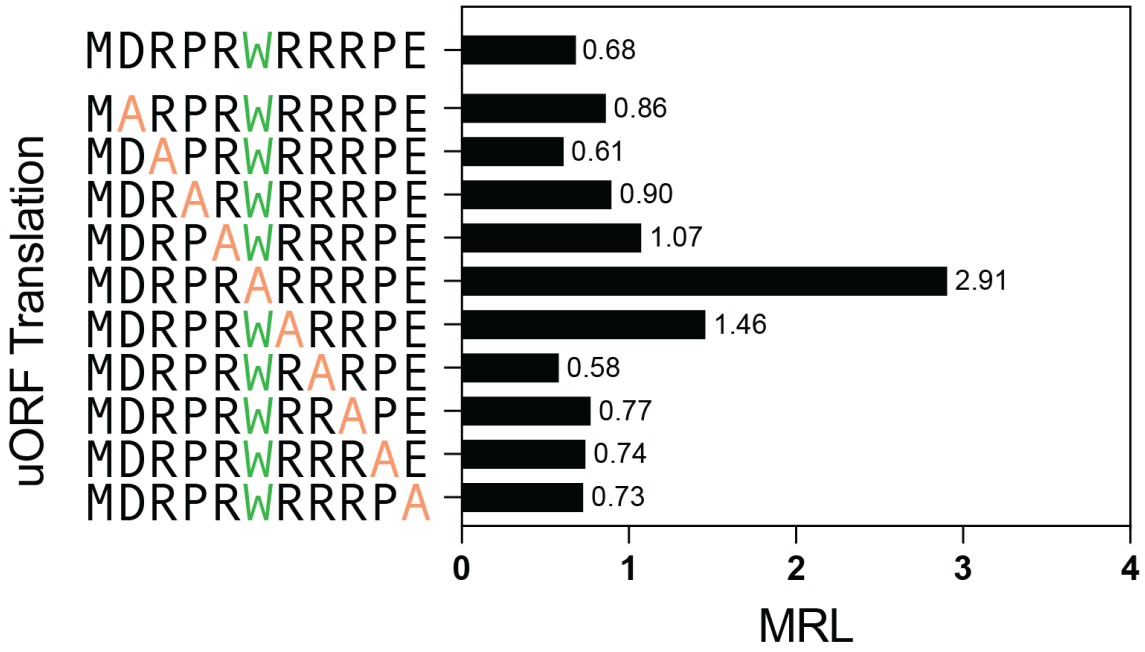


Figure 30. *In silico* Alanine scan to evaluate individual amino acid contributions

Machine learning MRL predictions of an *in silico* Alanine scan. The most-ihibitory MDRPRWRRRPE sequence (MRL = 0.68) was modified *in silico* to simulate an alanine scan, replacing each amino acid with an alanine in order to investigate amino acid criticality. Replacement of the W at position 5 clearly highlights that W at position 5 is a strong contributor towards the predicted inhibitory effect of this sequence.

3.4.6.5 Validation of the predicted motifs using the pDLR HOXA11 5′ UTR uORF system

The outcome of simulating two million uORFs and predicting their MRL resulted in the discovery two interesting motifs. The tryptophan at position 5 (5W) was found to be enriched in the sequences with the lowest MRL, while the RYG motif was found to be present

in the subset with highest MRL. The simulated uORF sequences were experimentally validated in the pDLR HOXA11 5′ UTR uORF reporter assay system. The results of this assay were then compared to the predicted MRL values.

These results again confirm the accurate prediction of increased MRL due to the ATG to TTG change in the *HOXA11* uORF initiation codon. Furthermore, the assay validates the inhibitory effect of 5W, which was found to be more inhibitory in comparison to the unchanged *HOXA11* WT uORF (**Figure 31**). The MRL predictions were not as accurate with the lowest MRL sequences. While one RYG motif molecule, L2_1RYG was found to be significantly less inhibitory than the *HOXA11* WT uORF, L5_1RYG was found to be more inhibitory. Thus, the predictions of the simulated uORFs with lower predicted MRLs were found to be less reliable. To further confirm the effect of 5W, its replacement with a 5A resulted in both higher predicted MRL and higher relative luciferase expression. However, adding the RYG motif in a uORF that contains the 5W motif did not rescue the inhibitory effect imparted by 5W (**Figure 31b**). This experiment highlights the potential of using trained machine learning models to extract actionable information, which can then be experimentally validated *in vitro*.

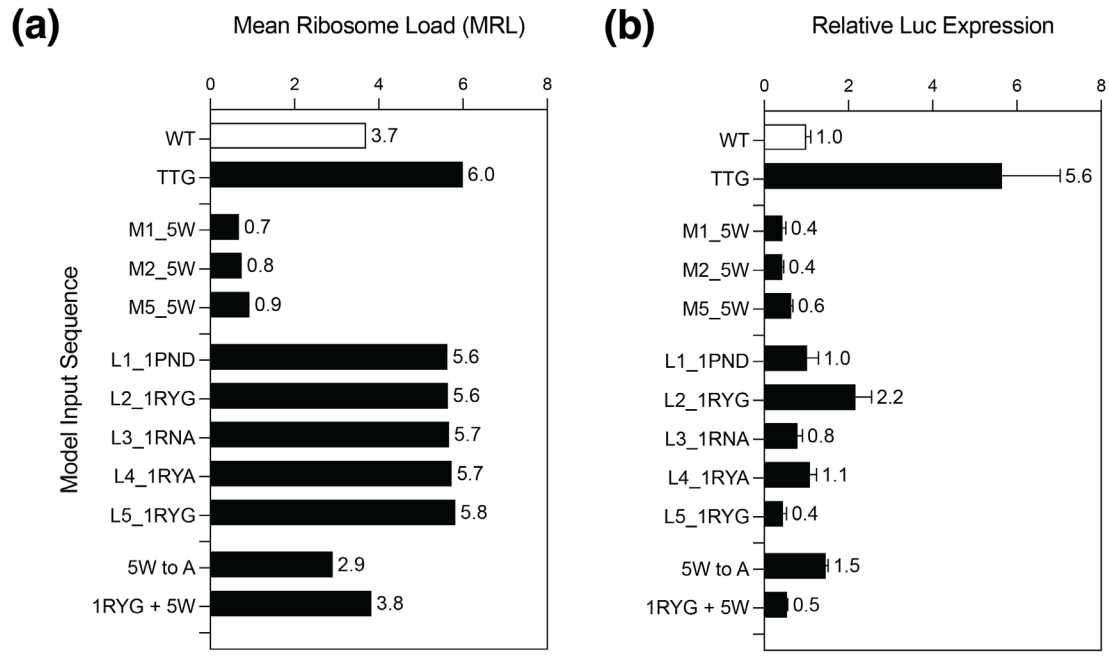


Figure 31. Validation of machine learning predicted MRLs

Predicted MRL and reporter assay validation of selected uORFs from the most (prefix M*N*_), and least (prefix L*N*_) inhibitory sequences. Suffix represents amino acid at a position (e.g. 1RYG means that the sequence begins with MRYG). (a) MRL predictions for simulated uORF sequences evaluated in the *HOXA11* 5′ UTR context (b) Dual-Glo luciferase assay data of simulated uORFs cloned in the pDLR *HOXA11* 5′ UTR uORF vector

3.4.6.6 Machine learning modeling of the effects of minimal uORFs with Optimus 5-Prime

Computational analysis of the Wang *et al.* proteomics data suggest that transcripts containing a single minimal uORFs are as inhibited as transcripts containing a uORF with a longer encoded sequence[132]. An analysis performed using the uORF Detective tool showed a strong initiation at minimal uORF in the *ATF4* gene. However, it was difficult to state conclusively that the minimal uORF was critical for the uORF regulatory architecture in the *ATF4* gene. Given that the Optimus 5-Prime model can predict MRL values at scale, the effects of introducing minimal uORFs in the 5′ UTRs of 6 genes (*SAMD11*, *ACOT11*, *ISG15*, *VAMP3*,

HES5, *HMX3*) that normally lack uORFs was investigated. The modeling was performed by introducing an AUGUAG uORF at every position in the 5′ UTR. Consistent with the proteomics analysis, the introduction of a minimal uORF in a 5′ UTR background that normally does not contain uORFs resulted in a decreased predicted MRL. Furthermore, some positional effects are evident as the slope of the regression line is negative in all cases, suggesting that a uORF introduction closer to the pORF is likely to result in stronger inhibition (**Figure 32**).

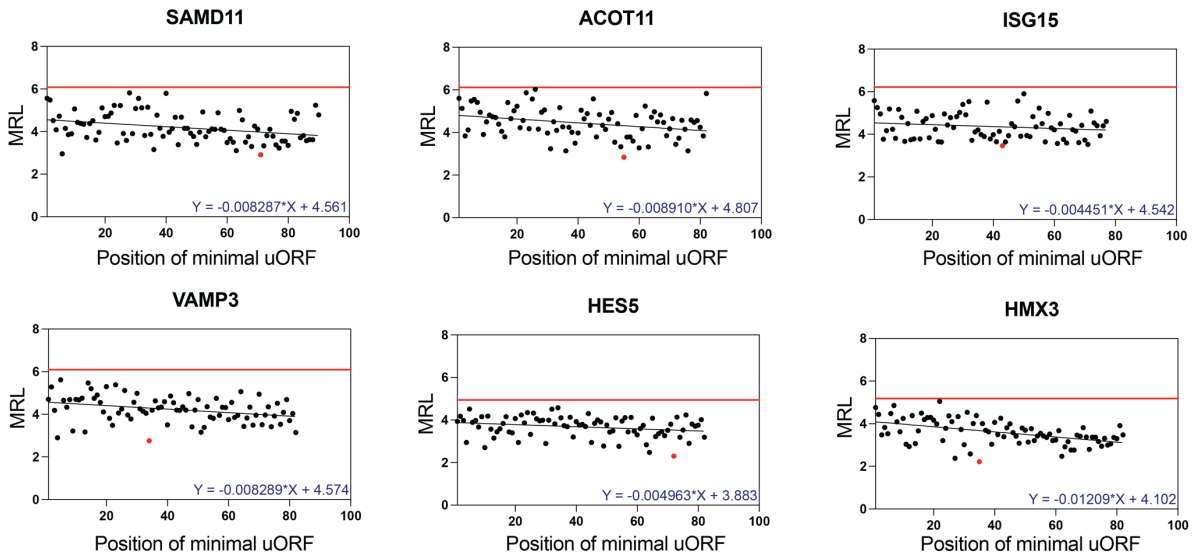


Figure 32. Predicting the effects of introducing a minimal uORF on MRL

An in silico MRL prediction and evaluation of the effects of introducing a minimal uORF in the 5′ UTR of six genes, whose wild type 5′ UTRs do not contain any uORFs. The minimal uORF AUGUAG was introduced at every position in the 5′ UTR of *SAMD11, ACOT11, ISG15, VAMP3, HES5 and HMX3* genes. The red line represents the predicted MRL of the unmodified 5′ UTR for each gene which does not contain any uORFs. Black dots represent predicted MRL values at each position where the minimal uORF was inserted. Red dot represents the lowest predicted MRL. Linear regression analysis was performed to evaluate the positional effects of the minimal uORF introduction.

3.5 Discussion

The research described in this section is an effort towards the understanding of the critical features that define the functional role of uORFs. For this purpose, two software tools were developed: the uORFeome database and uORF Detective, which represent a platform for future research and discovery of the mechanisms by which uORFs regulate translation. The uORFeome database is a comprehensive and current annotation of uORFs and their measurable features, while uORF Detective is an interactive web-based tool that facilitates the study of uORFs in context of ribosome profiling data.

The contribution of various uORF features towards uORF translational inhibition of the pORF was evaluated by intersecting information from the uORFeome database with publicly available transcriptomics and proteomics datasets. It was found that uORFs are depleted in highly expressed transcripts across 32 human tissues; suggesting a selection pressure against uORF inclusion in certain gene populations (**Figure 14A, B)**. Furthermore, analysis on protein expression data revealed that genes that contain uORFs in their 5′ UTRs have an overall lower protein expression in comparison to genes which lack uORFs in their 5′ UTRs (**Figure 15**). A more granular analysis on the influence of various uORF features revealed that uORF position and the number of uORFs in the 5′ UTR are key contributing factors to translational repression of the pORF (**Figure 16**). uORFs that are closer to the pORF were found to have a higher probability of being repressive. Interestingly, minimal uORFs, which do not encode a uORF peptide, were found to be repressive (**Figure 16c**). Artificial intelligence approaches also predicted minimal uORFs to be repressive (**Figure 32**). Deep learning experimental predictions were generated on millions of randomly generated uORFs, which led to the discovery of an inhibitory tryptophan residue at position 5 of the uORF-encoded peptide (**Figure 29a**). The machine learning predictions were confirmed experimentally using reporter

assays, highlighting the benefit of utilizing artificial intelligence techniques to solve biological problems.

3.5.1 uORFeome

Rapid technological advancements in the omics field are constantly improving versions of the genome, transcriptome and the proteome. Most recently, third generation sequencing, also known as long read sequencing (LR-Seq), was used to complete the last 8% of the human genome[337]. Similarly, the human transcriptome has grown to include more than 60,000 transcripts. The transcriptome undergoes frequent changes, and adjustments are often made in the 5′ UTR, which is typically underrepresented in mRNA sequencing data due to technical bias[338]. Keeping this in mind, the uORFeome was developed to pull the most recent transcriptome information at runtime, and therefore represents the most up-to-date annotation of uORFs and their measurable features.

Conceptually the uORFeome can be separated in 5 structural parts: 1. Processing of the annotated transcripts 2. Annotation of the uORFs on the genome and transcriptome 3. Measurements of features representing uORFs 4. Association with ribosome profiling data and analysis. 5. Linking to external databases. The uORFeome scripts pull data from the UCSC genome browser, the online Mendelian inheritance in man (OMIM) disease database, and the ClinGen dosage sensitivity databases. These association allow for the discovery of genes linked to pathogenic disorders that may benefit from uORF-mediated protein upregulation strategies.

3.5.2 Limitations of ribosome profiling studies

Ribosome profiling is a powerful technique that provides a snapshot of active translation on a transcriptome scale. However, several limitations of this technique have been

linked to the generation of false positives and false negative reads. Inherently, ribosome profiling relies on the concept of sequencing stretches of mRNA that are protected by the ribosome during endonuclease degradation[61,62,66,178,339]. The potential of mRNA to be engaged by RNA-binding proteins and non-coding RNAs may result in false positives arising from non-ribosomal protein-mRNA interactions[340]. Given that the ribosome protected fragments are on average 28nt long software solutions such as FLOSS (fragment length organization similarity score) can be used to filter out reads that are unlikely to arise from ribosome protected areas *in silico*[319]. Consequently, ribosome stacking may result in longer stretches of protected regions, which could be filtered out, resulting in false negatives.

The ribosome profiling technique uses compounds such as Cycloheximide to fix translating ribosomes in place, or Harringtonine and Lactimidomycin to fix initiating ribosomes at initiation codons. These compounds may introduce experimental distortions by altering local distributions of ribosomes[61,62,66].False negatives are likely to be prevalent in ribosome profiling data due to limited sequencing coverage or lack of data in tissues of interest[167]. For instance, we observed that some of the genes (e.g. *SCN1A*, discussed in Results 2) where we experimentally found uORFs to be strong regulators of translation had weak or no ribosome profiling data coverage.

3.5.3 uORF Detective

Despite its limitations, ribosome profiling data is a valuable resource that can be informative with regards to uORF activity in the transcriptome. The software solution uORF Detective, was developed to rapidly analyze uORF architecture on any transcript of interest. uORF Detective, uses information generated in the uORFeome to annotate uORFs and map aggregated ribosome profiling data. In order to improve efficiency, large-scale ribosome

profiling data was aggregated into global aggregate files for: footprints, initiating ribosomes, mRNA, 40S and a custom mRNA track.

uORF Detective uses the feature viewer JavaScript library to generate interactive websites for studying uORFs[331]. More generally, uORF detective improves on some of the limitations of using the genome browser for analyzing ribosome profiling data. Namely, uORF detective shows transcript level data, and is not complicated by the presence of introns or transcript directionality. Thus, uORF Detective is a more adequate tool to study ribosome profiling data, as this data is generated from mRNA, and not DNA.

3.5.4 Minimal uORFs

The bioinformatics analysis performed on the Wang *et al.* dataset suggests that uORFs with an initiation codon immediately followed by a stop codon are more inhibitory than protein coding uORFs(**Figure 16c)**[132]. This would suggest that initiation is the rate-limiting step, and the key event by which uORFs regulate translation at the pORF. The mechanism by which minimal uORFs inhibit translation are unclear. The uORF Detective analysis of the *ATF4* gene demonstrated that most initiating ribosomes are concentrated at a minimal uORF 0. However, when this observation was experimentally validated it was shown that the minimal uORF 0 had no obvious impact on the translation profile of the pORF. While more work is needed to establish the activity of the minimal uORF 0 under stress conditions, a three hypothetical uORF model was proposed for *ATF4*; one that includes initiation at the minimal uORF.

The dynamics of ribosome assembly followed by ribosome recycling, rather than elongation and termination have not been explored. In the absence of a post termination complex ribosome recycling may depend on eIF2D, MCT1 and DENR to destabilize the AUG - Met-tRNA$_i^{Met}$ pairing at the P-site[121,123,196]. It's quite possible that under such circumstances the 40S ribosomal subunit may even be able to remain associated with Met-tRNA$_i^{Met}$.

According to this hypothesis, minimal uORFs may be optimized for reinitiation. Several studies have associated uORF length with reinitiation potential, with longer uORFs >30 codons being experimentally shown to either reinitiate at lower rates or to not reinitiate[193,341]. The proposed mechanism for this decrease of reinitiation competency suggests that the ribosome would have more time to shed initiation factors during a longer elongation phase[342]. However, while several studies have found that uORF length is not a significant feature in context of translational repression of the pORF, we observed that uORFs encoded by ≤17 amino acids had a lower protein expression than longer uORFs (**Figure 16i)**[162,166]. An alternative way to initiate downstream of a minimal uORF would be if the minimal uORF is bypassed according to the leaky scanning model. Given that minimal uORF initiation codons are immediately followed by one of the three (UGA, UAG, UAA) termination codons, the +4 position is by default occupied by the U of the termination codon. This means that minimal uORFs initiation codons cannot be found in a strong Kozak sequence context. However, minimal uORFs may be found under an adequate Kozak context if the -3 position of the initiation codon is a purine. Both the uORF bypass, and optimized reinitiation hypotheses can't explain the observed lowered protein expression in genes which contain a single minimal uORF in their 5′ UTR (**Figure 16c**). Therefore, more work is needed to fully characterize the events that occur when the scanning ribosome encounters a minimal uORF.

3.5.5 Machine learning – potential and limitations

The work presented in this Chapter utilizes a publicly available pre-trained model which can be used as a resource for discovery. The Optimus 5-Prime model has some notable limitations that restrict its broad applicability on the transcriptome. Namely, the model was trained on sequences 25-100 nt in length, which means that any sequence that is tested with the

model must fit in that range. Thus, the model is only applicable to ~29% of the human transcriptome. Any testing of longer sequences requires truncation and possible loss of both information and accuracy of the predictions. This variable length model has an accuracy (r^2) between 0.74 and 0.84, with the performance on human sequences being $r^2 = 0.78$. Deep learning models can be further improved by training larger, high-quality datasets. A deep learning model with high accuracy can potentially be a transformative asset which can save significant resources and enable the testing of hypotheses at scale. The upfront cost for training such model is the generation of large-scale training data.

The experiments performed with the Optimus 5-Prime model were intriguing, as most of the predictions were corroborated by experimental validation with Dual-Glo luciferase assays. One limitation of this analysis is the unclear relationship between the dual glo luciferase relative luciferase expression and the predicted mean ribosome load value. However, positive correlation between these values, which are proxies for translation at the pORF is evident. A recently published model improves on the limitations of Optimus 5-Prime model and enables the testing of variable length sequences[343]. The ability of this model to accept input sequences longer than 100 nt enables the generation of MRL predictions on any 5′ UTR sequence.

3.5.6 Future prospects of computational uORF analyses

Large-scale omics datasets, and bioinformatics tools have facilitated the study of the genome and transcriptome at an unprecedented scale. However, while third generation LR-Seq technologies can now sequence entire isoforms, isoform level proteomics datasets are lacking. Protein sequencing is the next frontier of the omics field and carries the same potential that NGS enabled for biological research and therapeutic drug development [344]. Deep learning

models can help elucidate the non-obvious features that drive uORF activity and assist in understanding their hierarchy.

The computational methods presented in this Chapter demonstrate the annotation of putative uORFs under AUG and near-cognate initiation codons. However, while some of the uORF features were analyzed in context of large-scale proteomics data, it is not clear which uORFs in the transcriptome are active. An additional complication in assessing the criticality of uORFs is their behavior under changed translational dynamics such as stress. The understanding of this process may not only lead to the discovery of pathogenic uORF deregulation, but can facilitate a leap towards the understanding of the intricate mechanisms of the stress response.

3.5.7 Conclusions

Analysis performed on the uORFeome database highlights the prevalence of uORFs in the transcriptome and further reveals that most of the transcriptome contains near-cognate uORFs in the 5′ UTR; some of which are likely to be functional. Computational analysis of two large-scale transcriptome studies showed that highly expressed genes have a tendency to not include uORFs in their 5′ UTRs - a feature consistent across 32 human tissues.

uORF Detective analysis of the *ATF4* gene revealed a strong initiation at a previously unannotated minimal uORF 0. A new three-uORF model was proposed to explain the translational events in the 5′ UTR of the *ATF4* gene. However, more work is required to understand the regulatory mechanisms of minimal uORFs.

The regulatory potential of uORFs with regards to translation at the pORF is influenced by a complex set of interacting features including: the condition of the cell, the architecture of the 5′ UTR, the frame of the uORF relative to the pORF, the number of uORFs in the 5′ UTR, uORF position, uORF length, and termination codon usage. These findings were determined

by comparing the cumulative distributions of protein expression data for subsets of genes binned based on various uORF features. Significance of effect was determined by measuring the Kolmogorov-Smirnov distance (*D*), which represents the maximum distance between the empirical cumulative distribution functions (CDFs) at any point. Interestingly, the Kozak sequence and TIS Span were not found to be critical for the translational repression effect observed in the uORF-containing gene subset in comparison to the subset of genes that lack uORFs in their 5′ UTR.

The field of artificial intelligence has immense potential to revolutionize biology research and drug discovery. A deep learning prediction simulation of 2 million random uORFs led to the discovery of a novel inhibitory motif found in the uORF encoded sequence. The model accurately predicts the silencing effect, and subsequent higher mean ribosome load, when the uORF AUG initiation codon is placed under a less optimal UUG initiation codon.

In summary, the work presented in this Chapter lays out a computational foundation that can facilitate future uORF research, with the ultimate goal of opening new avenues for uORF-based therapeutic drug development.

4 Results 2: Modeling the repressive potential of uORFs in disease-relevant genes

4.1 Introduction

Mutations in the genome, including single nucleotide polymorphisms (SNPs), or larger deletions, insertions and duplications may lead to a loss of function (LOF) of a single copy of a gene. In some genes which are sensitive to these mutations, a single copy is insufficient to produce the necessary levels of mRNA translational template, required to support a normal phenotype. This group of genes, known as haploinsufficient (HI) genes, have recently gained interest as potential targets for therapeutic development, as loss of one functional copy in these genes has been associated with disease. More than 300 genes are known to exhibit a haploinsufficient phenotype in humans, many of which have been linked to disease conditions including tumorigenesis and neurodevelopmental disorders including Dravet syndrome (*SCN1A*), Sotos syndrome (*NSD1*), Smith-Magenis syndrome (*RAI1*), 22q11 deletion syndrome (22q11), hypertrophic cardiomyopathy (*MYBPC3*), and others[345,346]. Recent computational studies suggest that the number of haploinsufficient genes may be underestimated[347,348]. The most comprehensive dosage sensitivity database is maintained by the Clinical Genome (ClinGen) National Institute of Health (NIH) resource, and currently includes information on 2012 predicted haploinsufficient loci including 1503 genes and 509 regions, with 389 genes categorized as having sufficient evidence of haploinsufficiency[346].

In addition to HI genes, the expression of some genes may be downregulated as a result of disease. Upregulation of such genes may be employed as a disease modifying therapy. For example, decreased levels of the brain-derived neurotrophic factor (*BDNF*) which is critical for neuronal survival and growth, has been associated with neurodegenerative disease with

neuronal loss, including Alzheimer's disease, Parkinson's disease, multiple sclerosis (MS) and Huntington's disease[349].

Techniques that can selectively increase protein levels in native tissues are highly desirable, as most monogenic disease are caused by LOF. Several ON strategies have been used to restore gene LOF; most utilizing steric block ONs. The Duchenne muscular dystrophy approved drug Eteplirsen increases the translation of a partially functional Dystrophin by repairing the translational reading frame[246]. The Spinal muscular atrophy drug Nusinersen, promotes a critical exon inclusion to restore full length SMA protein levels[241,350]. Stoke Therapeutics have described an ON strategy called Targeted Augmentation of Nuclear Gene Output (TANGO), which aims to skip poison exons to prevent mRNA degradation by the NMD pathway[286,287]. An orthogonal approach to the ON strategies is protein replacement by mRNA delivery, which enabled the development of mRNA vaccines for the COVID-19 pandemic. Namely, while mRNA as a protein replacement therapy has faced significant intracellular delivery challenges, the delivered amounts were found to be sufficient to expose the immune system to the mRNA-encoded spike protein[351,352].

This section focuses on evaluating the consequences of silencing uORFs in the 5′ UTR as a novel strategy for increasing protein translation at the pORF. The overarching aim of the work presented in this section is to investigate the feasibility of targeting uORFs for the purpose of protein upregulation, understand the dynamic range of uORF derepression, and investigate if targeting opportunities are possible in genes with complex multi-uORF 5′ UTR architectures. For this purpose, multiple therapeutically relevant gene panels were used, representing a broad spectrum of conditions including cancer, neurodegenerative, and neuromuscular disorders.

4.2 Hypothesis and aims

I hypothesize that translation initiation at uORFs is a driving factor for the translational inhibition of pORFs, which can be abrogated by silencing the uORF initiation codon through deliberate mutation. I further hypothesize that the alteration of uORF architecture can lead to upregulation of expression at the pORF and propose this mechanism as a strategy for therapeutic upregulation of gene expression.

This hypothesis was tested by:

- Modeling the derepression of translation at the pORF by weakening uORF initiation codon in bona fide disease-relevant genes.
- Examining the possibility of upregulation in disease-relevant genes which do not contain uORFs in their 5′ UTR sequence.
- Evaluating the possibility of upregulation in genes with complex multi-uORF 5′ UTR
- Changing the positional and Kozak contexts of uORFs to gain insights into the contribution of these features to uORF activity.

4.3 Materials and methods

4.3.1 Selection of an oncogene and tumor suppressor gene panel

The uORFeome database was used to obtain a set of genes containing uORFs. This set was intersected with the Cancer Gene Consensus database to obtain a list of 8,778 transcripts. Transcripts where the uORF overlaps the pORF were filtered out, resulting in a set of 5,923 transcripts. From this set, 130 tumor suppressor genes (TSG) and oncogenes were identified, and 11 genes of interest (GOI) were selected for subsequent screening. This panel is represented by the oncogenes (*SMO, MAP2K2, KDR* and *JUN),* fusion (*HOXA11*), and the oncogene and TSG (*FOXL2).* The *SRY* and *RNASEH1* genes were added to this panel as their regulation by uORFs has been previously published.

A secondary expansion panel from the same list of genes includes the fusion (*HOXA9*), the oncogene (*IKBKB*), oncogene and fusion (*KDM5A*, *BRD3*), oncogene and TSG (*MAP3K1*), and TSG (*NCOR1*). The 5′ UTR sequences of these GOI were each cloned into the pDLR GOI 5′ UTR vector by Gibson assembly. The pDLR GOI 5′ UTR utilizes RLuc as a universal pORF and monitors the translation levels of the pORF to evaluate the regulatory effect of the GOI 5′ UTR.

4.3.2 Deletion walk strategy

The genes *DMD*, *CISD2* and *MYBPC3*, do not contain uORFs in their 5′ UTR sequence, but upregulation of these genes could be beneficial for treating their associated disease symptoms. For example, the upregulation of dystrophin translation may be beneficial as a combinatorial drug therapy with exon skipping therapeutics, while *CISD2* and *MYBPC3* haploinsufficiency is linked with non-alcoholic fatty liver disease (NAFLD) and hypertrophic

cardiomyopathy respectively[246,353,354]. For these uORF-lacking genes, inhibitory sequence search was performed by cloning 5′ UTR inserts with deleted regions. Regions ~25 nucleotides in length were removed from the 5′ UTR designs, such that X1 represents the 5′ UTR with the first 25 nt removed. X2 would exclude a region from positions 26-50 (**Figure 33**)

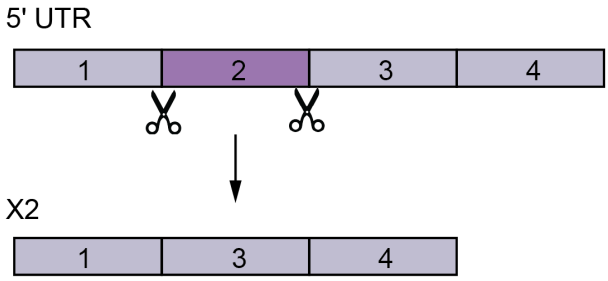


Figure 33. Deletion walk strategy for the detection of non-uORF inhibitory elements in the 5′ UTR

A schematic representation of a deletion walk strategy employed to search for sequence regions that may be inhibitory for translation of the pORF. 5′ UTR sequences lacking uORFs were evaluated for the presence of inhibitory elements by iteratively removing 25 nt regions from the 5′ UTR. The truncated sequences were cloned into the pDLR 5′ UTR vectors.

4.4 Results

4.4.1 Translational upregulation of the pORF by weakening uORF initiation codon in a panel of oncogenes and tumor suppressor genes

For the initial evaluation, the choice was made to screen a panel of genes implicated in cancer that carry 1 uORF in their 5′ UTR sequences. Proto-oncogenes and tumor-suppressors are particularly interesting in context of uORFs, as uORF-affecting mutations may result in malignant phenotypes. It has previously been discussed by Wethmar *et al.* that this can happen in one of two ways: (i) a mutation that weakens the initiation codon in a uORF in the 5′ UTR of a proto-oncogene can result in upregulation of expression of the pORF, (ii) a mutation creating a uORF in the 5′ UTR of a tumor-suppressor gene can inhibit translation at the pORF[318]. A precedent for the occurrence of the latter has been established in the *CDKN2A* gene, where a *de novo* uORF introducing mutation promotes the development of melanoma, due to inhibition of translation at the pORF[355]. More generally, the idea of upregulating TSGs by targeting uORFs is enticing, as it can have profound implications in the development of cancer therapeutics.

The original gene panel includes *SMO, MAP2K2, KDR, JUN, HOXA11, FOXL2, RNASEH1* and the *SRY* genes. *SRY* and *KDR* were used as controls as they were previously screened by a similar approach by Calvo *et al.* , while *RNASEH1* was included due to recent publications by Ionis Pharmaceuticals demonstrating uORF targeting by ONs[216,217]. The strategy of silencing uORFs by introducing an AUG to UUG mutation of the initiation codon has been previously demonstrated and is based on previous knowledge with regards to the general efficiency of initiation at various near cognate codons (NCCs)[82]. It is known that initiation at NCCs occurs with lower efficiency than initiation at the canonical AUG codon[82]. The efficiency of initiation of the UUG NCC has been evaluated to be very low in HEK293T

cells, representing a 98% reduction relative to the translation initiation at the AUG codon[334]. This finding has been confirmed in a more recent analysis of transcript initiation site efficiencies in humans, where the UUG NCC initiation efficiency was found to be the lowest of all (N)UG NCCs[322]. Therefore, the substitution of the uORF canonical AUG initiation codon with UUG can be thought of as "silencing" of initiation at the uORF, as such modification is predicted to reduce initiation efficiency by ~95%.

It was observed that uORF silencing can lead to a significant (~2-8 fold) increase of RLuc translation in several genes including *HOXA11*, *MAP2K2*, *RNASEH1* and *SMO* (**Figure 34**). Significant variability in RLuc translation across the various 5′ UTRs was observed, highlighting the critical nature of 5′ UTRs in post-transcriptional regulation. Consistent with previous observations, significant changes in the mRNA levels of the RLuc transcripts were not observed, indicating that the regulatory effect is post-transcriptional in nature (not shown)[162]. Notably, the derepression effect was not significant in some genes, suggesting that not all uORFs act as potent repressors at the pORF.

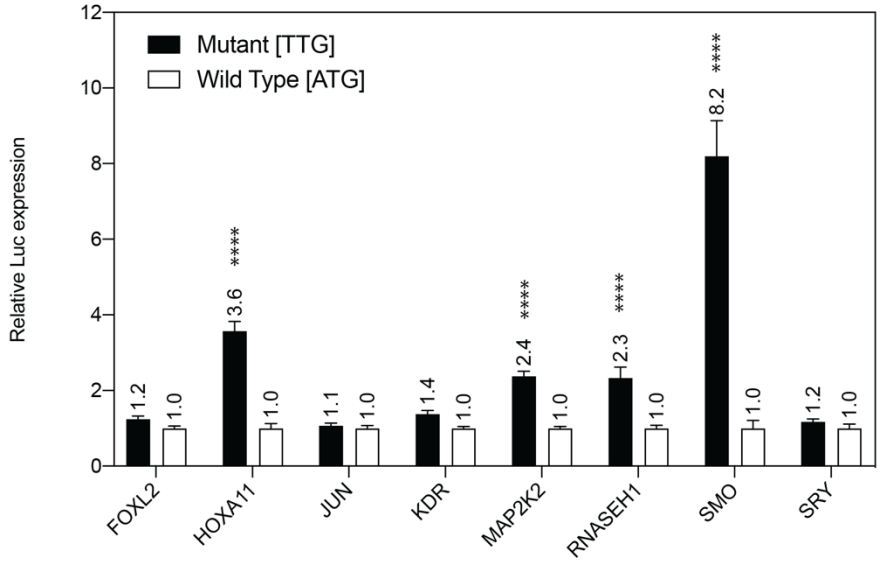


Figure 34. Evaluation of the regulatory potential of 5′ UTRs containing wild-type or mutated uORFs in a Dual-Glo Luciferase reporter assay

Plasmid DNA of the pDLR GOI 5′ UTR reporter vectors, with cloned 5′ UTRs containing a single uORF with either a wild type - ATG initiation codon (white), or a mutated - TTG initiation codon (black), was transfected in HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant reporter construct, normalized to its associated wild type construct. Error bars represent ±SD of ≥4 biological replicates. Statistical significance was evaluated by a Student's *t*-test relative to the corresponding wild-type control. Asterisks indicate significant difference compared to the wild-type 5′ UTR (****$P \leq 0.0001$).

To evaluate the sensitivity of the report assay approach, the data were normalized to the empty pDLR CTRL 5′ UTR vector. The results suggest that the reporter assay is sensitive enough to show a 6-fold increase or 10-fold decrease of relative luciferase expression in comparison to the empty vector control. This conclusively highlights the importance of the 5′ UTR leader sequence in regulating the levels of protein expression (**Figure 35**).

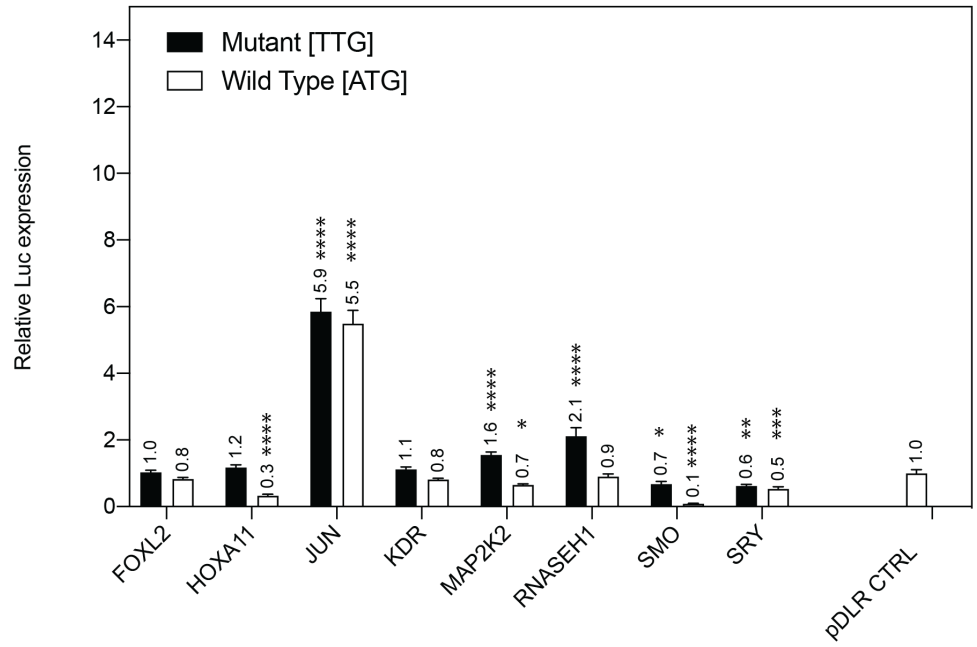


Figure 35. Evaluation of the sensitivity of the Dual-Glo Luciferase reporter assay

Plasmid DNA of the pDLR GOI 5′ UTR reporter vectors, with cloned 5′ UTRs containing a single uORF with either a wild type - ATG initiation codon (white), or a mutated - TTG initiation codon (black), was transfected in HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant reporter construct, normalized to the empty pDLR GOI 5′ UTR reporter vector. Error bars represent ±SD of ≥4 biological replicates. Statistical significance was evaluated by a Student's *t*-test relative to the corresponding wild-type control. Asterisks indicate significant difference compared to the wild-type 5′ UTR (*$P \leq 0.05$, **$P \leq 0.01$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

The initial cancer related GOI panel was supplemented with additional genes from the original search. The *HOXA11* gene was rescreened as a positive control, to evaluate assay variability. Significant increase in RLuc translation was observed in all genes, most prominently in *HOXA11*, *NCOR1*, *KDM5A,* and *HOXA9* (**Figure 36**).

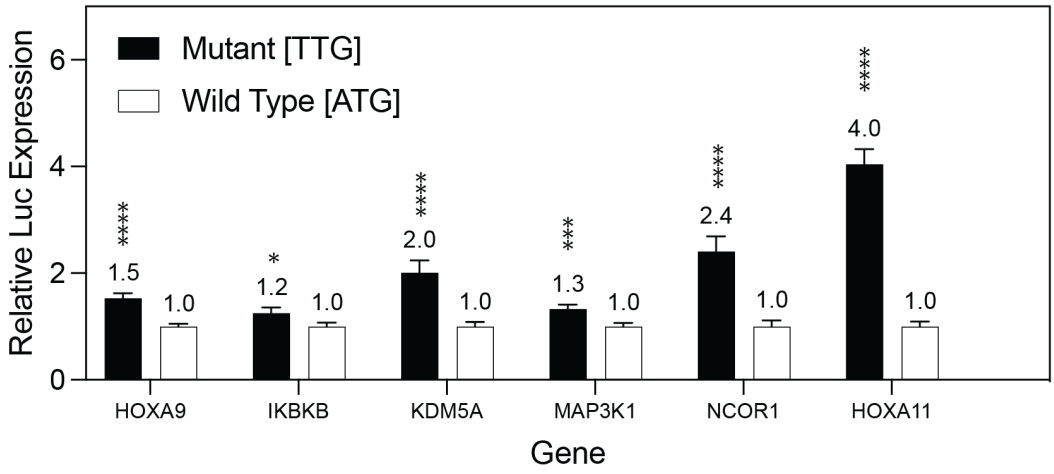


Figure 36. Evaluation of the regulatory potential of *bona fide* 5′ UTRs containing wild-type or mutated uORFs in a Dual-Glo Luciferase reporter assay

Plasmid DNA of the pDLR GOI 5′ UTR reporter vectors, with cloned 5′ UTRs containing a single uORF with either a wild type - ATG initiation codon (white), or a mutated - TTG initiation codon (black), was transfected in HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant reporter construct, normalized to its associated wild type construct. Error bars represent ±SD of ≥4 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (*$P \leq 0.05$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

4.4.2 Translational upregulation of the pORF by weakening uORF initiation codon in neuromuscular disease

Technological advancements in the omics field have enabled the ability to genetically define a growing number of neuromuscular diseases (NMD). However, despite our increased knowledge in the mechanisms of disease, a significant unmet need for treatment persists. NMD impacts approximately 14,000,000 people globally, with the average prevalence of all inherited NMD estimated to be 37-49.9 per 100,000[356]. Harnessing the regulatory nature of uORFs presents a new and unexplored opportunity, which may be of great benefit for NMD drug

development. Thus, encouraged by the observations from the cancer panel experiments, an additional panel of genes associated with neurological/neuromuscular pathology was evaluated. This GOI in this panel include two transcript variants of the *BDNF* gene, *C9orf72* which is found to be a major cause of both frontotemporal dementia (FTD) and amyotrophic lateral sclerosis (ALS), the glial cell derived neurotrophic factor (*GDNF)* associated with Hirschsprung disease – a common congenital disorder resulting in intestinal obstruction and lethality, Huntingin (*HTT*) which in a repeat expansion form is the cause for Huntington's disease (HD), and finally the sodium voltage-gated channel alpha subunit 1 (*SCN1A*), the haploinsufficiency of which is the cause of Dravet syndrome. In addition to these GOI the *GATA2* transcription factor was included as, *GATA2* haploinsufficiency has been associated with numerous conditions including immunodeficiency, myelodysplastic syndrome, and vascular/lymphatic problems[357,358].

In this panel, the selected GOIs were not limited to including a single uORF, which allowed for a first look into the regulatory nature of complex 5′ UTR uORF architectures. The 5′ UTRs of the GOIs (*BDNF, GDNF, and SCN1A*) contain multiple uORFS. To evaluate the total repressive effect of their uORFs, all of the uORF initiation codons were mutated from AUG to UUG. Interestingly, as predicted in the bioinformatics analysis (**Figure 16a**), genes with the most uORFs (*BDNF, GDNF, and SCN1A*) exhibited greater derepression when their uORFs were silenced (**Figure 37**). However, the overall observation was that expression of the pORF in all the tested genes in the NMD panel are subject to strong uORF-mediated regulation, revealing potential for uORF targeting.

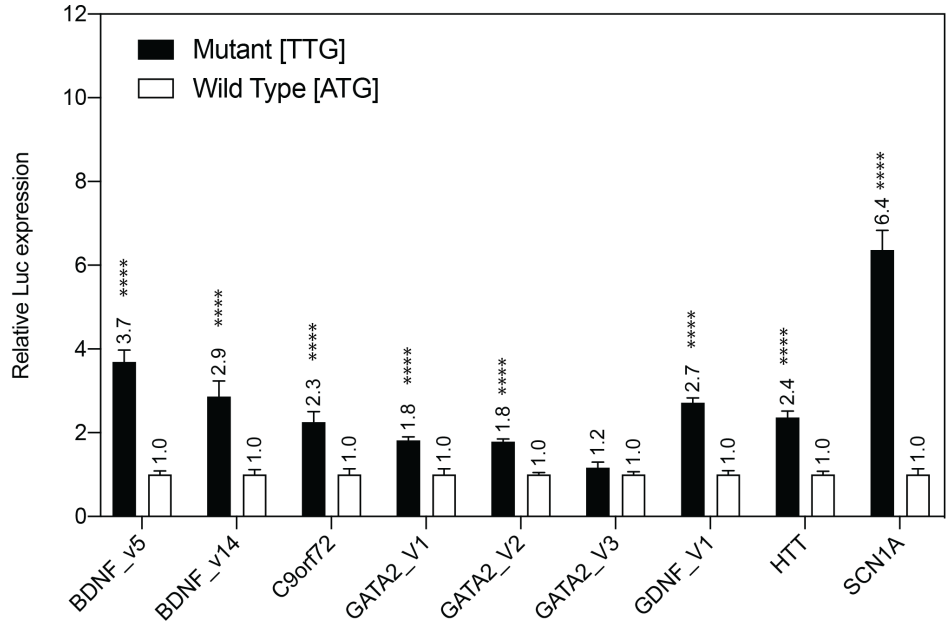


Figure 37. Evaluation of the regulatory potential of *bona fide* 5′ UTRs containing wild-type or mutated uORFs in neuromuscular disease GOI

Plasmid DNA of the pDLR GOI 5′ UTR reporter vectors, with cloned 5′ UTRs containing a single or multiple uORFs with either a wild type - ATG initiation codons (white), or a mutated - TTG initiation codons (black), was transfected in HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant reporter construct, normalized to its associated wild type construct. For several GOIs, including (*BDNF*, *C9ord72*, *GATA2* and *GDNF*) specific variants were selected, labeled according to their NCBI nomenclature. For GOIs with multiple uORFs (*SCN1A* and *BDNF*) all uORF initiation codons were mutated. Error bars represent $\pm$SD of $\geq$4 biological replicates. Statistical significance was evaluated by a Student's *t*-test relative to the corresponding wild-type control. Asterisks indicate significant difference compared to the wild-type 5′ UTR (****$P \leq 0.0001$).

4.4.3 Mechanism of translation repression in genes with multiple uORFs

uORFs are predominantly found in multi-uORF complex 5′ UTR arrangements. Understanding which uORFs contribute most to the repressive effect in complex multi-uORF 5′ UTRs is critical for identifying putative therapeutic targets. Using a similar strategy as in the *ATF4* example discussed in the Results 1 section, 3 genes *BRD3* (NM_007371), *BDNF*

(NM_170733, NM_001143814), and *SCN1A* (NM_006920) containing multiple uORFs in their 5′ UTRs, were used to explore the dynamics of uORF regulation in complex 5′ UTR architectures. The *BRD3* gene was selected from the cancer gene panel, the *BDNF* gene is associated with multiple neurodegenerative disorders, and *SCN1A* haploinsufficiency is the cause of Dravet syndrome. While the number of possible permutations is too large to screen (e.g. for *SCN1A* there are 2^{12} permutations), directional cumulative mutants from both the beginning and end of the leader sequence were generated to gain insights into the repressive potential of individual uORFs.

The *BRD3* gene is characterized by two uORFs in the same frame as the pORF. Interestingly, these uORFs share the same TGA stop codon (**Figure 38**). Individual uORF silencing, in this gene suggests that uORF 1 is more repressive, despite the Ribo-Seq prediction that uORF 2 should be targeted. However, this is likely due to lack of data close to the 5′ cap. When both uORF 1 and uORF 2 are disrupted an additive derepression effect is observed. The 2.4 fold increase in RLuc pORF translation suggest that even in multiple uORF architectures, some uORFs may still be targetable and can be therapeutically relevant.

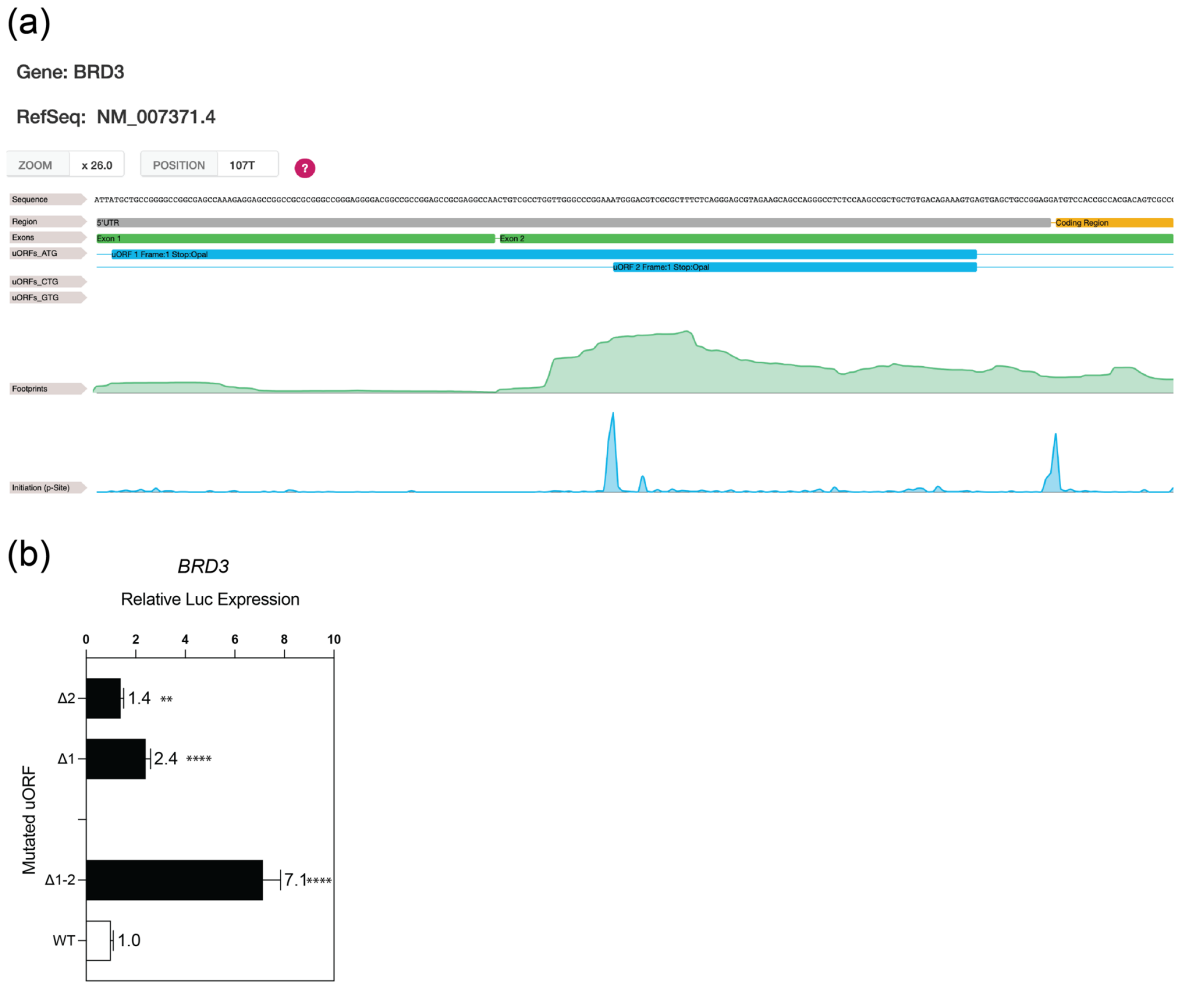


Figure 38. Evaluation of the regulatory potential of individual uORFs in the *BRD3*

(a) A uORF Detective map for *BRD3* shows two overlapping uORFs which share a stop codon, with uORF 2 showing a stronger initiation signal and a visible initiation signal at the pORF (b) For each uORF, the relative luciferase expression is shown for reporter constructs where the uORF initiation codon has been changed to TTG (black, also denoted by Δ, where Δ*x* means only the initiation of uORF *x* was changed to TTG), normalized to that of the construct containing the parent wild-type 5′ UTRs (white). Error bars represent ±SD of ≥4 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (**$P \leq 0.01$, ****$P \leq 0.0001$).

To further validate this, a similar study was performed for two variants of the *BDNF* gene. These results suggest that the maximum repression effect is cumulative and is not a result of translation repression by any one uORF in particular, as none of the individually mutated uORFs (Δx) was not derepressed to the level of the completely mutated uORFs (**Figure 39** Δ1-5, and **Figure 39b** Δ1-3). However, it should be noted that the levels of derepression by mutating individual uORFs was higher in *BDNF* (NM_001143814); the variant which has lower number of uORFs (3 vs 5) (**Figure 39b**). Genes with multiple uORFs may be more repressed because their 40S ribosomal subunits have a higher likelihood to initiate at repressive uORFs. The independent contributions of uORF repression were examined by reintroducing the wild-type initiation codon in the fully mutated background (**Figure 39a** Δ1-5, and **Figure 39b** Δ1-3). Directional, cumulative, and sequential mutants were generated from the 5′ and 3′ of the UTR in order to evaluate the effects of the combined repressive effects and distance to the pORF. As expected, mutating more uORFs results in elevated translation of the RLuc pORF, highlighting the cumulative inhibition in multi-uORF 5′ UTR architectures.

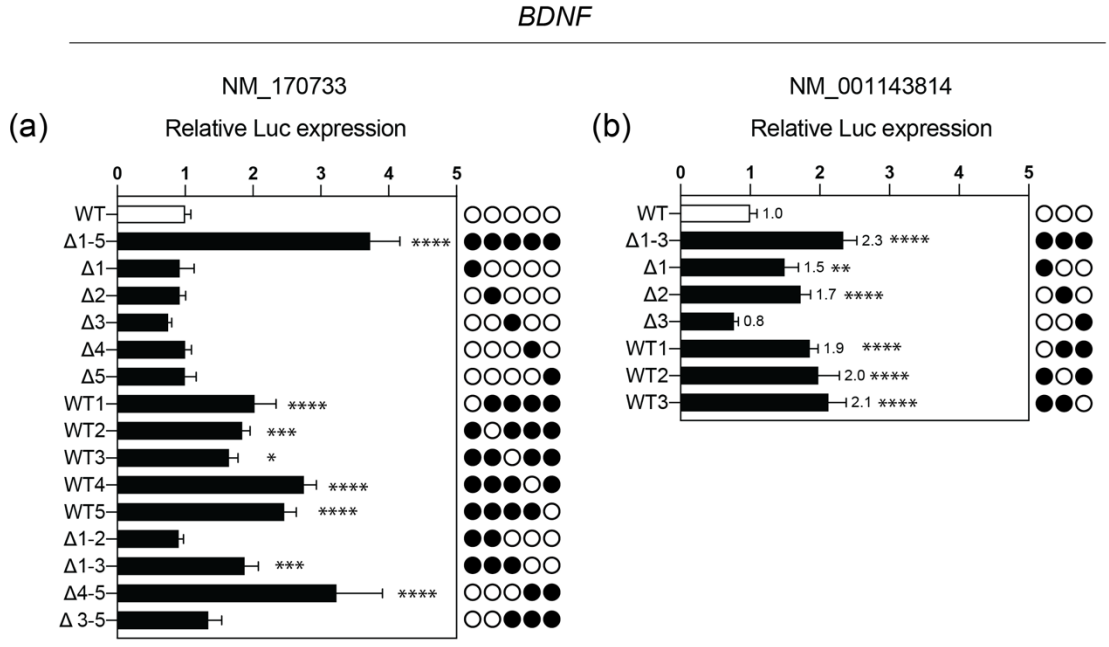


Figure 39. Dual luciferase assay evaluating the individual and combinatorial effects of uORFs in the multi-uORF 5′ UTRs of two splice variants of the *BDNF* gene.

A reporter assay experiment evaluating the effects of uORF silencing permutations in two variants of the *BDNF* gene. Plasmid DNA of the pDLR *BDNF* 5′ UTR reporter vectors, with cloned 5′ UTRs containing multiple uORFs with either wild type - ATG initiation codons (white circles), or mutated - TTG initiation codons (black circles), were transfected in HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant reporter construct permutation (black), normalized to the wild type construct (black). The positional changes are shown with white and black circles, where white circles represent unchanged uORF initiation codons, and black circles represent an ATG to TTG change of the uORF initiation codon. Two *BDNF* transcript variants were evaluated (a) NM_170733 which contains five uORFs in its 5′ UTR, and (b) NM_001143814 which contains three uORFs in its 5′ UTR. Error bars represent ±SD of ≥4 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (*$P \leq 0.05$, **$P \leq 0.01$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

Haploinsufficiency of the gene *SCN1A* is the cause of Dravet syndrome - a rare severe epileptic encephalopathy, characterized by seizures that are refractory well to anticonvulsant medications. Dravet syndrome is a devastating pediatric disorder with poor prognosis. The prevalence rate has been estimated to be 1 in 20,000 to 1 in 40,000 children[359]. The 5′ UTR architecture of the *SCN1A* gene is highly complex and includes 12 uORFs distributed across its sequence. A key motivation to model the effect of uORFs on the pORF expression in SCN1A was to investigate the possibility of finding regulatory uORFs in GOIs with complex 5′ UTRs. While testing all possible combinations was not feasible, as it would require the generation of 2^{12} constructs, the impact of each individual uORF was investigated, along with directional aggregated mutants starting from both ends of the 5′ UTR. In all cases, individual uORFs were silenced by changing their initiation codons from AUG to UUG.

It was observed that the expression of the *SCN1A* pORF, as modeled by RLuc in the luciferase constructs, is highly repressed by its constitutive uORFs in the 5′ UTR, as silencing of these uORFs results in a significantly higher (16.3-fold) translation of the pORF. Interestingly, silencing individual uORFs could not rescue this repression, suggesting redundancy and additive effects. However, it was observed that the individual silencing of the last three uORFs proximal to the pORF result in a 1.8-2.9-fold increase in RLuc signal, suggesting that these three uORFs may represent potential targeting regions for upregulating *SCN1A* (**Figure 40**, Δ10, Δ11, and Δ12). Interestingly, these 3 uORFs overlap the pORF initiation codon and terminate in the CDS.

Next, to investigate the contributions of individual uORFs, each uORF was separately kept unmodified in a completely silenced background. The inhibitory potential of the last three uORFs (uORF 10, uORF 11, and uORF 12) was confirmed in the mutated background (**Figure 40**, WT10, WT11, and WT12). It was also observed that uORF 1 and uORF 5 are active and

suppressive uORFs (**Figure 40**, WT1, WT5). Given that uORF 1 and uORF 5 are distal from the pORF, it may be possible that they promote a more efficient ribosome dissociation post translation. Finally, progressive silencing of uORF initiation codons from both ends of the 5′ UTR clearly highlights the increased repressive potential of uORFs proximal to the pORF.

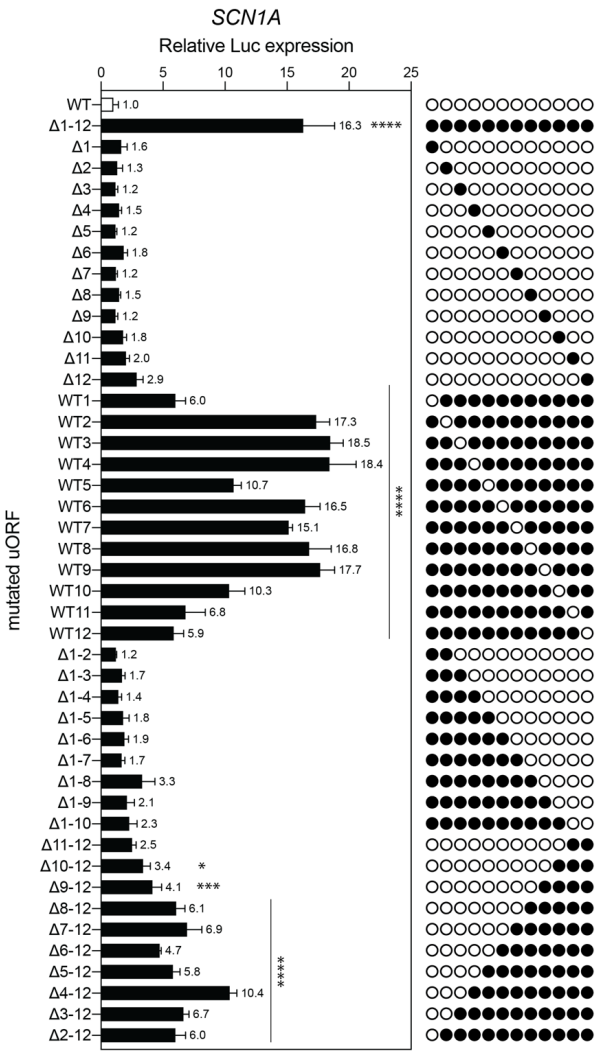


Figure 40. Dual luciferase assay evaluating the individual and combinatorial effects of uORFs in the multi-uORF 5′ UTRs of two splice variants of the *SCN1A* gene.

A reporter assay experiment evaluating the effects of uORF silencing permutations of the *SCN1A* gene. Plasmid DNA of the pDLR *SCN1A* 5′ UTR reporter vectors, with cloned 5′ UTRs containing multiple uORFs with either wild type - ATG initiation codons (white circles) or mutated - TTG initiation codons (black circles), were transfected in HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant reporter construct, normalized to the wild type construct. The positional changes are shown with white and black circles, where white circles represent unchanged uORF initiation codons, and black circles represent an ATG to TTG change of the uORF initiation codon. The relative luciferase expression is shown for reporter constructs permutations (black), normalized to that of the wild-type construct containing unchanged uORFs (white). Error bars represent ±SD of ≥8 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (*$P \leq 0.05$, **$P \leq 0.01$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

4.4.4 A deletion walk for investigating the presence of inhibitory elements in the 5′ UTRs of therapeutically-relevant GOIs which lack uORFs

The initial results reveal uORFs as promising targets for therapeutic GOIs in cancer and neuromuscular disease. However, 51.5% of all genes lack AUG-initiating uORFs in their 5′ UTR sequence. Upregulation of expression of the pORF in many of these genes could be of significant therapeutic value. Three genes (*DMD, CISD2,* and *MYBPC3*) were selected whose phenotypes have been directly associated with haploinsufficiency.

The *DMD* gene, is the largest known human gene, represented by 79 exons and spanning a region of more than 2,200 kb codes for the protein called dystrophin. Mutations in the *DMD* gene, predominantly frame-shifting deletion mutations that lead to the absence or LOF of dystrophin, are the cause of Duchenne Muscular Dystrophy (DMD), a rare neuro-muscular X-linked disorder characterized by progressive muscle degeneration[246,254,360]. A milder form of DMD, Becker Muscular Dystrophy (BMD) is caused by in-frame deletion mutations[361]. DMD predominantly affects boys, and its global prevalence has been estimated to be 7.1 cases per 100,000 males[360].

Haploinsufficiency of the membrane protein CISD2 has been recently associated in the manifestation of nonalcoholic fatty liver disease (NAFLD), nonalcoholic steatohepatitis (NASH), and human hepatocellular carcinoma (HCC)[354,362]. Moreover, Cisd2 has been implicated in aging in mice as it has been found that Cisd2 levels progressively decrease with aging and knockout Cisd2 mice have a shortened lifespan[363]. Conversely, stable high levels of Cisd2 in mice has been shown to prolong lifespan[364]. *CISD2* LOF is also implicated in Wolfram syndrome 2 (WFS2), an autosomal recessive neurodegenerative disorder associated with a shortened lifespan[365].

The cardiac myosin binding protein C (*MYBPC3*) is found in cardiac cells and its function is associated with stabilizing thick filaments of the cardiac muscle sarcomeres[366].

Mutations that cause haploinsufficiency of the *MYBPC3* gene have been linked to hypertrophic cardiomyopathy (HCM) – the most common inherited cardiac disease[353,367].

Given that the GOIs (*DMD, CISD2, MYBPC3*) do not have uORFs in their leader sequences a ~25 nucleotide deletion walk strategy was used to investigate if these genes are repressed by some other *cis* sequence motifs, such as miRNA or RNA binding protein (RBP) binding sites. Interestingly, no increase in relative luciferase levels was observed (**Figure 41**). Moreover, in some cases (e.g. *DMD* X2, *CISD2* X2 and X4, and *MYBPC3* X2) the deletions were found to be inhibitory for the translation of the RLuc pORF. This suggests that the leader sequences of these genes (*DMD*, *CISD2*, and *MYBPC3*) do not contain repressive sequences.

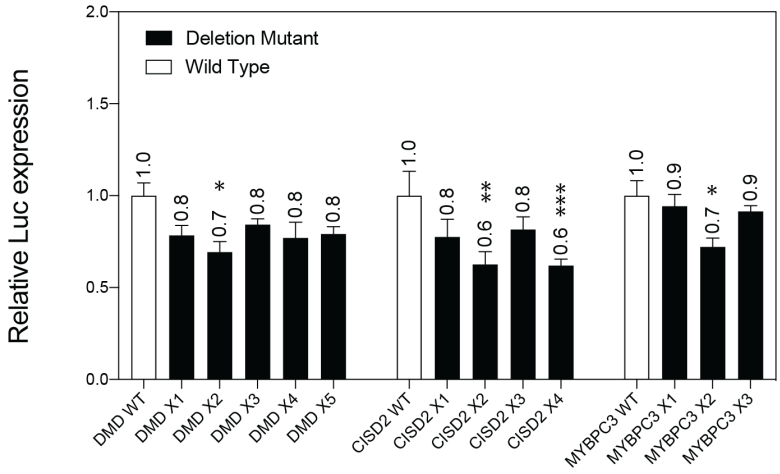


Figure 41. A deletion walk strategy to search for inhibitory elements in the 5′ UTR in disease relevant GOIs

A reporter assay experiment evaluating the effects of 5′ UTR sequence deletions, for the purpose of finding elements that may be inhibitory for translation of the pORF in the *DMD*, *CISD2* and *MYBPC3* genes. Plasmid DNA of the pDLR GOI 5′ UTR reporter vectors, with cloned 5′ UTRs containing deletions was transfected into HEK293T cells. Luciferase signal was measured after 24 hours. The relative luciferase expression is shown for each mutant deletion reporter construct, normalized to the full-length wild type construct. The 5′ UTRs are modified by deleting ~25nt region sequences as described in the Methods section. Statistical significance was evaluated by one-way ANOVA with

Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (*$P \leq 0.05$, **$P \leq 0.01$, ***$P \leq 0.001$).

4.4.5 Investigating the effect of uORF position on translation at the pORF

The increase of protein levels produced by the survivor motor neuron 2 (*SMN2*) gene, and utrophin (*UTRN*) can have significant therapeutic benefit for the treatment of Spinal Muscular Atrophy (SMA) or DMD. Namely, the steric block ON drug nusinersen targets *SMN2* transcripts to promote the production of full-length SMN protein[350]. An increase in the translation of nusinersen-repaired transcripts in SMA may lead to improved phenotypes. Similarly, *UTRN* is a homologue of the *DMD* gene. Due to their significant similarity, it is believed that *UTRN* can rescue function in *DMD* deficient muscle[368]. Thus, the upregulation of *UTRN* may be of benefit in the treatment of DMD.

When the 5′ UTRs of *SMN2* and *UTRN* were assessed for activity with the pDLR GOI 5′ UTR reporter assay no derepression of the RLuc pORF was observed, suggesting that the uORFs in these genes are not functionally repressive and cannot be used as targets for upregulation of the pORF (**Figure 42a**). Interestingly, in both genes the uORF is located very close to the start of the 5′ UTR sequence (**Figure 42b**). This result is consistent with the proteomics data analysis discussed in Chapter 1 as well as previously published data that suggests that uORFs that are distal from the pORF are less likely to be repressive[162]. Next, to further elucidate the impact of position and initiation context on uORF activity, the uORF and uORF Kozak sequence elements of the 5′ UTRs of *SMN2* and *HOXA11* genes were investigated in various positional architectures. There are two key differences between the uORFs of *SMN2* and *HOXA11*, other than the uORF sequence: (i) the *SMN2* uORF is positioned near the 5′ cap, at a distal position from the pORF, while the *HOXA11* uORF is positioned at a proximal position to the pORF, and (ii) the *HOXA11* uORF is under an adequate Kozak sequence context (AGAAUCAUGU), while *SMN2* is under a weak Kozak sequence context (CCACAAAUGU).

In the first two examples, it was observed that the *HOXA11* uORF significantly repressed the RLuc pORF, while the SMN2 uORF is inactive.

To assess if uORF position was the reason for the SMN2 uORF lack of repression, a domain swap experiment was performed, where the *SMN2* uORF was moved to a medial (in the middle of the 5′ UTR) and proximal (at the end of the 5′ UTR) position relative to the pORF in the *SMN2* 5′ UTR background. It was observed that the proximal position showed inhibition of the pORF (**Figure 42c**). When the *SMN2* uORF is used to replace the *HOXA11* uORF in the *HOXA11* 5′ UTR background, while using the *SMN2* Kozak sequence the derepression is similar. However, when the *SMN2* uORF is placed in the *HOXA11* 5′ UTR background under the *HOXA11* Kozak sequence there is a significant repression of the pORF, that is abolished when the AUG codon is replaced to a UUG (**Figure 42c**). This suggests that the Kozak sequence context is likely an important feature for uORF initiation. It should be noted that in the genome-wide proteomics analysis in the Results 1 section the initiation context was not found to be a critical contributing factor to uORF repression. However, significance of the Kozak sequence has been demonstrated in a computational analysis of mouse proteomics data. Although, with a *P*-value of 0.04 it was the weakest uORF feature reported[162]. It is possible that the Kozak sequence becomes an important contributing factor once more influential factors, such as positional context, are optimized for repression. The effect of uORF position on translation of the pORF, was further confirmed by placing the active *HOXA11* uORF in the *SMN2* 5′ UTR background at the *SMN2* uORF position near the 5′ cap. It was observed that with either adequate or weak Kozak context, there was no significant repression on the RLuc pORF (**Figure 42c**).

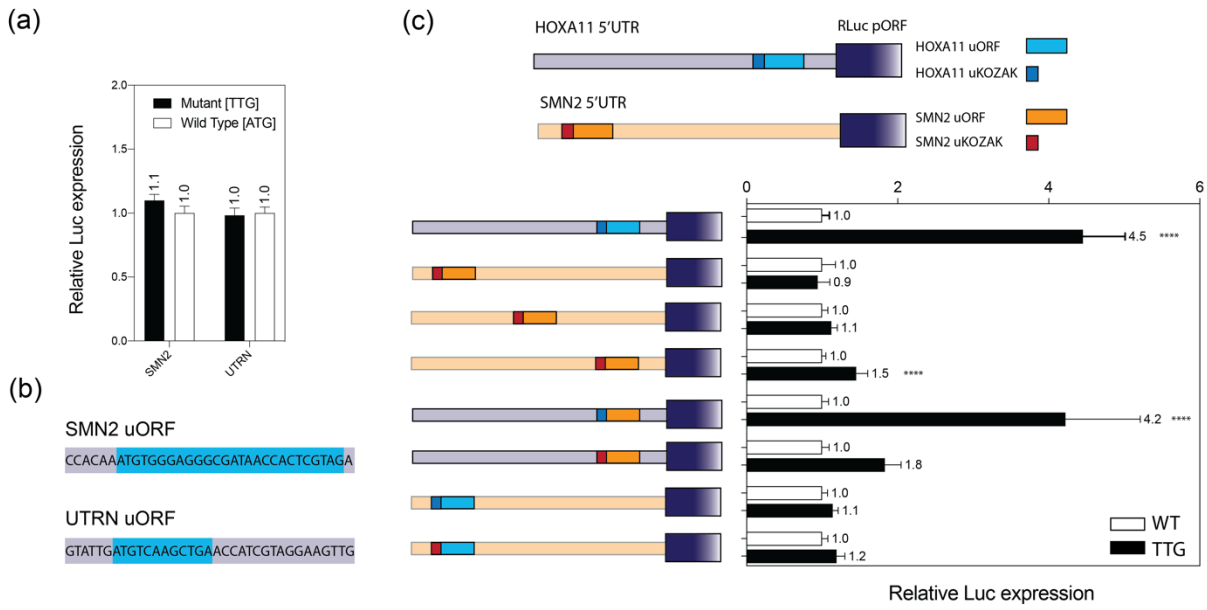


Figure 42. uORF activity is dependent on position and initiation context

(a) A reporter assay experiment evaluating the effects of uORF silencing by modifying the uORF initiation codon from ATG to TTG in the *SMN2* and *UTRN* genes. For the experiments shown in (a) and (c) wild type or mutated 5′ UTRs were cloned in the pDLR GOI 5′ UTR reporter vector and plasmid DNA was transfected into HEK293T cells. Dual-Glo luciferase assay was performed as described in the methods (b) The uORFs of the *SMN2* and *UTRN* genes, shown in blue, are located at the beginning of their 5′ UTRs near the cap structure. (c) Evaluating the importance of position and initiation context by repositioning uORFs. Depicted in the legend (top) are the *HOXA11* 5′ UTR (grey) with the *HOXA11* uORF (light blue), and its corresponding Kozak sequence (dark blue). The *SMN2* 5′ UTR is depicted in light orange, with its *SMN2* uORF (orange), and its corresponding Kozak sequence (dark orange). Below, each combination is represented with an ATG uORF initiation codon (white) and a mutated uORF initiation codon TTG (black). Relative luciferase expression is shown for the following combinations: the wild-type and mutant *HOXA11* uORF in its native background, the native *SMN2* as well as its relocation by sliding it to a medial and proximal positions relative to the pORF, the *SMN2* uORF placed in the *HOXA11* 5′ UTR context under the *HOXA11* Kozak context or its own Kozak context. The *HOXA11* uORF placed in the *SMN2* 5′ UTR context with its own Kozak sequence, or the Kozak sequence of the *SMN2* uORF. Error bars represent ±SD of ≥8 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the wild-type 5′ UTR (****$P \leq 0.0001$).

4.5 Discussion

4.5.1 uORF as potential targets for neuromuscular diseases and cancer

Neuromuscular diseases are group of congenital or inherited disorders that involve injury, deterioration, or dysfunction of peripheral nerves or muscle tissues[369,370]. Despite increased ability to genetically characterize neuromuscular diseases, therapeutic options are lacking for most conditions. Hereditary neuromuscular disorders are rare, with their prevalence in Europe is estimated to be between ~33-112/100,000[371]. Given that many neuromuscular disorders are progressive in nature, many patients can suffer increasingly debilitating symptoms for many years. Therefore, the treatment of neuromuscular disorders represents a significant burden on healthcare systems.

For many neuromuscular disorders, the diversity of their genetic mutations limits the therapeutic availability to a fraction of the patients suffering from the disorder. For example, the drug ataluren, licensed in Europe for the treatment of DMD in patients with nonsense mutations, is applicable to only 10-15% of DMD cases. While eteplirsen, an exon skipping ON developed by Sarepta Therapeutics is only amenable to mutation architectures that can be corrected by skipping of exon 51; represented by approximately 13-15% of DMD cases[360]. Casimersen, which similarly to eteplirsen corrects mutations through skipping of exon 45, is applicable to 13% of the DMD patient population. Together, targeting exons 8, 44, 45, 50, 51, 52, 53, and 55 in DMD is estimated to be of benefit to 47% of patients, highlighting the unmet need for therapeutic development[361]. Despite the availability of approved ON treatment options, the levels of partially functional dystrophin restored by treatment has been controversially low[246]. Orthogonal treatments that can improve translation of the rescued dystrophin could be transformative. Unfortunately, it was not possible to model uORF-regulation in the *DMD* gene as it does not contain uORFs in its 5′ UTR. Instead, a sequence

deletion search was performed as an attempt to find any potentially inhibitory elements in the 5′ UTR. Given that no inhibitory elements were found, utrophin (*UTRN*), a *DMD* paralogue was explored. It has been proposed that *UTRN* may be able to compensate for *DMD* LOF[368,372]. However, the positional aspect of the *UTRN* uORF makes it inactive in context of regulating translation at the pORF. Additional observations in *ATF4* (Results 1) and *SMN2* genes, suggest that uORFs close to the 5′ cap may not be as inhibitory as those found proximal to the pORF.

Interestingly significant upregulation of the pORF was observed in genes implicated in numerous neurological disorders including *HTT, C9orf72, BDNF,* and *SCN1A.* While treatment options targeting these genes are currently in development, it is important to note that targeting uORFs is an orthogonal technique to existing splice correction and allele-specific knockdown strategies and could therefore potentially amplify translation of the repaired or wild-type transcript if used as a combinatorial therapeutic. For example, the allele-specific targeting of *HTT* or *C9orf72* toxic gain of function alleles may result in haploinsufficiency, which may be resolved by increased translation of the wild-type, healthy allele. Similarly, the TANGO platform used by Stoke Pharmaceuticals to upregulate SCN1A results in the increase of mRNA levels[286,287]. If protein levels are insufficient, uORF-mediated upregulation could improve the translatability of the restored mRNA.

Significant upregulation of translation was also observed in two cancer panel datasets. Cancer is a disease caused by the deregulation of two types of genes: tumor suppressor genes and proto-oncogenes. It was observed that both categories can be regulated by uORFs. For instance, the AUG to UUG mutation which silences the uORF of the tumor suppressor *NCOR1* increases the translation at the RLuc pORF by 2.4-fold. This suggests that uORFs can be attractive targets for either upregulating the expression of tumor suppressor genes, or attenuating the expression of proto-oncogenes.

4.5.2 uORF features and their contribution towards uORF activity

The ability to predict uORF activity is critical for identifying potential therapeutic targets. Given that more than 50% of all transcripts have complex uORF architectures, it is necessary to understand both which uORFs are more repressive, and the features that promote uORF activity. The work presented in this section validates several computationally predicted features including position, Kozak context, number of uORFs.

Due to the established initiation competence of the scanning ribosome assembled at the cap structure, it is likely that ribosomes initiate at the first uORF initiation codon. However, depending on the distance from the uORF termination codon to the next initiation codon, ribosomes may or may not reinitiate. The fact that upregulation at the pORF was not observed at *SMN2* or *UTRN*, and the fact that *SMN2* is active when it's moved to a proximal position to the pORF, suggests two possibilities: (i) uORF reinitiation may be a very efficient process and 40S ribosome are more likely to stay engaged with the mRNA than dissociate post termination, or (ii) uORFs that are too close to the 5′ cap may not initiate efficiently. It has been proposed that the GTP hydrolysis of eIF2•GTP, may occur during scanning, with the phosphate P_i remaining engaged through its interaction with eIF1 until an initiation codon is reached. This mechanism is facilitated by the eIF2-specific GTPase-activating protein eIF5[20]. Thus, if eIF2•GTP is not hydrolyzed before an initiation codon is reached, the scanning ribosome may be more likely to bypass cap-proximal initiation codons.

Interestingly, an adequate *HOXA11* Kozak sequence was found to improve uORF repression of the translocated *SMN2* uORF in the *HOXA11* 5′ UTR background. However, it has been a marginally significant feature in published bioinformatics analysis on the mouse proteome, and was found to not be a significant feature in a much larger proteomics analysis from the human proteome shown in the Results 1 section[162]. It is possible that the global

bioinformatics analysis underestimates the importance of the Kozak sequence, which may be dominated by stronger features such as uORF position within the 5′ UTR.

Finally, three key observations were made by exploring multi-uORF complex 5′ UTR architectures in the *BRD3*, *BDNF*, and *SCN1A* genes: (i) consistent with observations from analyzing the human proteome, and published bioinformatics analysis in the mouse proteome, 5′ UTRs with multiple uORFs are under a stronger translational repression which depends on the number of uORFs, (ii) even in such cases, silencing of some uORFs leads to a significant upregulation of the pORF; opening the possibility for therapeutic targeting even in more complex 5′ UTR architectures, and (iii) the positional aspect is a dominating feature, as progressive silencing of uORFs proximal to the pORF lead to derepression. The silencing of the uORFs proximal to the pORF increases the intercistronic distance to promote reinitiation after the translation of medially positioned uORFs, but also rescues ribosomes from initiating at overlapping uORFs which would normally bypass the pORF initiation codon.

4.5.3 Future prospects of uORF applications to therapeutic development

4.5.3.1 Towards more predictive ribosome profiling datasets

The *BRD3* example highlights an important aspect of uORF Detective analysis, relevant to the interpretation of ribosome profiling data. While it appears that most of the initiation signal comes from uORF 2, the data availability at uORF 1 is sparse. This phenomenon complicates the interpretation of Ribo-Seq data and may be especially problematic in the global aggregate approaches, as aggregation amplifies the bias. Due to technical limitations of standard RNA sequencing assays the exons at the 5′ UTR of transcripts are underrepresented in sequencing datasets. This assay limitation may be further amplified in the more complex Ribo-Seq protocols. Therefore, to improve reliability of Ribo-Seq datasets, individual datasets must eventually be sufficient to show true signal without dataset aggregation. Given the

implications of variable uORF usage under stress conditions, Ribo-Seq datasets under such conditions can be an invaluable asset to completely understand the dynamics of uORF translation.

4.5.3.2 Harnessing uORFs for targeted gene regulation

Given their diverse regulatory nature under different conditions, harnessing uORFs is an attractive avenue for targeted gene regulation. The results presented in this chapter demonstrate that silencing of uORF initiation codons may be a viable strategy for restoring function at the pORF. This concept can be applied across different conditions where protein restoration may be of benefit including haploinsufficiency disorders, tumor suppressor gene upregulation, or disease modifier upregulation. Going in the opposite direction, mutations that introduce a uORF in the 5′ UTR can be used to attenuate translation at the pORF. For example, AUA positions can be targeted for an AUG conversion with single base editing techniques. These editing techniques can involve permanent modifications on a DNA level by using CRISPR approaches, or transient approaches such as ADAR-mediated RNA editing. Single guide CRISPR strategies have recently been applied to targeting uORFs in plants and human cells[373-375]. However, while the possibility of developing a cure with genome editing is enticing, more work is required to fully understand the potential consequences of aberrant editing at unanticipated regions. Transient approaches that target RNA are conceptually a safer option until CRISPR platforms are improved. Considering this, RNA targeting ON methods are likely to coexist with DNA editing for the foreseeable future as exemplified by SMA and DMD. Notably, the specificity of CRISPR platforms has increased significantly with the development of CRISPR Prime[376].

uORF architectures can be used in the design of gene replacement therapies where inducible or tunable expression is desirable. For instance, the uORF termination codon to

pORF distance, or uORF Kozak sequence features can be used to fine-tune the translational activity at the pORF. Recently, it has been discussed that continuous AAV9-mediated SMN transgene expression may be related to toxicity, highlighting the necessity for improvements in transgene expression systems. Perhaps the most intriguing aspect of uORFs is their ability to regulate expression at the pORF differentially during stress conditions. uORF architectures that promote expression only under stress condition may be an attractive strategy for inducible transgene expression in disease conditions.

4.5.4 Conclusions

The research presented in this chapter highlights the function of uORFs as potent regulators of translation at the pORF. The ability of uORFs to attenuate the expression of the pORF was assessed by cloning 5′ UTRs from multiple genes implicated in cancer and neuromuscular disease. uORFs were silenced by mutating their initiation codon from AUG to UUG, placing the uORF under a suboptimal near cognate codon. It has been previously shown that such change decreases initiation by as much as 98% in comparison to the canonical AUG initiation codon. Several predictions from the bioinformatics analysis in the Results 1 section were confirmed. It was observed that genes with complex uORF architectures containing multiple uORFs are under strong uORF repression. However, even in these cases the silencing of some uORFs can lead to significant upregulation of translation of the pORF, suggesting that pORF upregulation can be accomplished by targeting a single uORF even in genes with complex 5′ UTRs. In the most complex case that was examined, which features 12 uORFs in the 5′ UTR, the most potent uORFs were proximal to the pORF. In fact, their termination codon was found to overlap the CDS of the pORF. To further support the positional aspect predicted by the bioinformatics results, uORFs close to the m7G cap were found to have no effect on translation at the pORF. The translocation of the inactive *SMN2* uORF into the

HOXA11 5′ UTR pORF proximal position restored its inhibitory potential. This was further improved by strengthening the Kozak sequence. More work is required to completely understand the hierarchy and influence of various uORF features on uORF function. These results establish that the expression of multiple disease-relevant genes is regulated by uORFs, and supports the further exploration of uORFs as potential targets for therapeutic development.

5 Results 3: Targeting uORFs with steric block and RNA editing oligonucleotide technologies

5.1 Introduction

The technological revolution in the genomics and proteomics fields has enabled the genetic and mechanistic characterization of numerous diseases. However, while gene variants and pathologic proteins have been clearly associated with disease, a growing number of therapeutically relevant targets have been considered intractable from a perspective of established small molecule or biologics drug development. The limited applicability of these two classes of therapeutics in the treatment of many conditions have branded multiple therapeutic targets as "undruggable"[377]. After nearly four decades of development, a new class of therapeutics called oligonucleotides (ONs), has started delivering on the promise of "drugging the undruggable"[219,377]. ONs are modified nucleic acid polymers, capable of engaging a broad spectrum of molecules across the extended central dogma including DNA, RNA, Protein, and derivatives of posttranslational processes[218,219]. Pharmacokinetic improvements driven by advancements in ON chemistry have unlocked the potential of some ON modalities, which has translated to recent successes in the clinic[218,219,313].

The broader class of ONs encompasses several diverse groups of molecules categorized based on their structure, chemical architecture, and mechanism of action. The most prevalent ONs are antisense oligonucleotides (ASOs) - synthetic single stranded ONs which exert their activity through Watson-Crick base pairing to a cognate region on the RNA. Based on their chemical modification architecture, ASOs can function through one of three mechanisms: (i) in a "gapmer" form, where the center of the ASO is comprised of DNA, while the "wings" are chemically modified, ASOs can elicit endonucleolytic cleavage of its complementary target

RNA by recruiting RNase H, (ii) in their fully chemically modified form which lacks a DNA gap, ASOs serve as steric blocks which have been utilized to regulate splicing events, and (iii) in a similar steric block form ASOs have been used as inhibitors of translation[218-220,255,266,378].

Another class of ONs with recent translation in the clinic are small interfering RNAs (siRNA). Unlike ASOs, siRNAs ONs are 20-25 bp long double-stranded RNAs (dsRNAs), which act through the endogenous RNA interference (RNAi) pathway to elicit cleavage and subsequently degradation of mRNA. Depending on design, siRNAs may require processing by the Dicer enzyme, or may directly engage the RNA-induced silencing complex (RISC) – a multiprotein assembly which includes catalytic enzyme Argonaute 2 (Ago2). When this occurs, one of the strands, also known as the guide strand, gets loaded into RISC and serves to recognize a cognate site on the mRNA to initiate cleavage by Ago2. The remaining strand, also known as a passenger strand, is cleaved and discarded during the RISC loading step. siRNA ONs target specific mRNA loci to promote mRNA degradation mediated by the RISC factor and are therefore suitable for the downregulation of pathogenic proteins[144,313]. siRNAs share the same pathway with another class of molecules called microRNAs (miRNAs). Endogenous miRNA biogenesis begins as a longer stem-loop dsRNA, but after two processing steps mediated by the enzymes Drosha and Dicer, the processed double-stranded RNA product resembles an siRNA, and can load into RISC. In addition to mRNA cleavage promoted by interactions with cognate sequences, miRNA can also regulate mRNA via near-cognate sequence interactions, which promotes decapping and deadenylation of target mRNA. The near-cognate interactions enable individual miRNAs to regulate larger groups of transcripts[137,145]. ASOs, siRNAs, and miRNAs all act on RNA via Watson-Crick base pairing. A fourth category of ONs called aptamers diverge from this paradigm and are capable of engaging proteins and their posttranslational modifications through three-dimensional structure interactions[219,379].

The majority of approved ON drugs to date belong to the ASO and siRNA categories and have been concentrated on gene silencing and splice modulation modalities. However, the idea of harnessing ONs for gene activation has been gaining traction due to the large number of conditions that can benefit from upregulating therapeutics. The first association with ONs and protein upregulation was discussed by Wahlestedt in 2006 in a review article proposing that siRNA can be used to target natural antisense and noncoding RNA transcripts to elevate gene expression[380]. Later in that same year, Li *et al.* targeted the promoter of E-cadherin with dsRNAs and observed an Ago2-dependent induction of expression[381]. While this led to coining of the terms RNA activation (RNAa) and small activating RNA (saRNA), the existence of an RNAa pathway is still contentious as the mechanism of gene upregulation remains poorly defined. Recently, saRNA were found to recruit the RNA-induced transcriptional activation (RITA) complex (comprised of AGO2, CTR9, and DDX5), to activate the transcription of the tumor suppressor C/EBPα in hepatocellular carcinoma[382]. MiNA Therapeutics is currently evaluating the safety and efficacy of MTL-CEBPA, the first saRNA to be tested in the clinic. The Phase II study is currently recruiting patients to evaluate synergistic effects with the kinase inhibitor Sorafenib[383-386].

Several additional strategies for protein upregulation have been described. A recent platform developed by Stoke Therapeutics called targeted augmentation of nuclear gene output (TANGO), aims to increase the pool of translatable mRNA, by rescuing premature stop codon-containing isoforms from degradation by the NMD pathway. Stoke Therapeutics have reported promising results from their Phase 1/2a MONARCH study of STK-001, which utilizes a splice switching ASO evaluated for the treatment of Dravet Syndrome – a genetic epileptic encephalopathy caused by haploinsufficiency of the *SCN1A* gene[265,286,387-390].

An additional ON upregulation approach targets inhibitory sequences, such as natural antisense transcripts (NATs) such as long non-coding RNA, or endogenous small RNAs which

are known to downregulate gene expression[391]. Targeting these silencing elements can restore function in inhibited transcripts, as has been previously demonstrated in *BDNF* (Parkinson's disease), *UBE3A* (Angelman syndrome) and *SCN1A* (Dravet syndrome), among others[392-394].

Recently, scientist at Ionis Pharmaceuticals published a paper reporting that uORF initiation codons can be targeted by short steric block ASOs to elicit a translational upregulation effect at the pORF[216,217]. Liang *et al.* observed an unusual dose-dependent bell curve effect of upregulation with several ON chemistries but fall short of conclusively demonstrating a mechanism for their observed effects. The research presented in this chapter reevaluates these oligonucleotide designs. In addition to this, this chapter presents an effort to examine the ability to edit uORF AUG initiation codons using a novel class of ONs that recruit the Adenosine Deaminase Acting on RNA (ADAR) enzymes, capable of catalyzing the deamination of Adenosine to Inosine. Inosine can form two hydrogen bonds with Cytosine, and as such is recognized by the ribosome as Guanosine. This modification would result in silencing of the uORF, as initiation at the GUG near-cognate codon is estimated to be 9% of that of the AUG codon initiation[82,334]. This strategy is used to demonstrate the first RNA editing of a uORF, for the purpose of protein upregulation.

5.2 Hypothesis and aims

I hypothesize that protein upregulation can be achieved by targeting upstream open reading frames with oligonucleotide technologies.

This hypothesis was tested by:

- Evaluating the ability of steric blocking uORF-targeting oligonucleotides to elicit protein upregulation
- Examining the ability to mutate AUG codons with three different RNA editing oligonucleotide triggers.
- Demonstrating uORF RNA editing proof of concept.

5.3 Materials and methods

5.3.1 Transfection of editing antisense oligonucleotides in the presence of human interferon alpha A (IFNα)

HeLa, HEK293T, and SF8628 cells were cultured in complete media. On the day before transfection, cells were trypsinized, counted and seeded in 96-well plates in 100 µl of complete media. Due to differences in cell division and cell size cells were seeded as follows (HEK293T: 2.5×10^4, SF8628/HeLa: 1.5×10^4 per well). The following day, the cell media was removed, and replaced with 130 µl of complete media without Pen/Strep, containing human interferon alpha (IFNα) (Sigma-Aldrich). 0.3ul of Lipofectamine RNAiMax per well was used to prepare a 20 µl transfection mixture of OptiMEM-diluted oligonucleotides. 20 nM and 40 nM final oligonucleotide concentration were tested in 150 µl volume per well. RNA was extracted 48 hours after transfection for analysis.

5.3.2 Cotransfection of Dual-Glo luciferase 5′ UTR model plasmids and steric block ASOs with Lipofectamine 2000

pDLR GOI Plasmid DNA of *RNASEH1* and *HOXA11* mutant and wild type vectors was prepared at a concentration of 20 ng/µl, while ONs were prepared at 2 µM starting stock concentrations in nuclease-free water. HEK293T cells were seeded in opaque 96-well tissue culture plates in 80 µl of complete media without Pen/Strep at a density of 3×10^4 per well. 20 µl transfection solution AB was prepared by mixing solution A (0.5µl of Lipofectamine 2000 + 9.5 µl of OptiMEM) with solution B (plasmid DNA + ON) and incubating at room temperature for 15 minutes. Solution B was prepared such that the final transfected plasmid DNA concentration was 100 ng per well, while ON concentrations were variable depending on experiment design. Plates were incubated for 24 hours before dual luciferase assay readout.

5.3.3 Dual transfection of pDLR GOI 5′ UTR plasmid DNA and steric block ASOs with Lipofectamine 3000 and RNAiMAX

HEK293T cells were seeded in 6-well plates at a density of 0.5×10^6 per well. The following day, each well was forward transfected with 0.6 µg of pDLR GOI 5′ UTR plasmid DNA using 6 µl of Lipofectamine 3000 per well, following the manufacturer's protocol. After 24 hours the transfection media was replaced with complete media without Pen/Strep and cells were left to recover for ~2 hours. Then, cells were detached, counted and reseeded in opaque 96-well tissue culture plates in 80 µl of complete media without Pen/Strep at a cell density of 3×10^4 per well, for a subsequent RNAiMax reverse transfection with steric block ASOs. 20 µl transfection solution AB was prepared by mixing solution A (0.3µl of RNAiMax + 9.7 µl of OptiMEM) with solution B (OptiMEM + ON) and incubating at room temperature for 15 minutes. The transfection solution was then mixed with the cells in suspension. Plates were incubated for 24 hours before dual luciferase assay readout.

5.3.4 Co-transfection of ADAR p110 and 5′ UTR model plasmids, and sequential transfection of editing antisense oligonucleotides in the presence of human interferon alpha A (IFNα)

HEK293T cells were seeded in 12-well tissue culture plates at a density of 0.35×10^6 per well in 1 ml of complete media. The cells were left to attach overnight in a humidified incubator set at a temperature of 37°C, with 5% CO_2. On the following day, the complete media was gently removed and replaced with 1 ml of complete media without Pen/Strep – containing IFNα (Sigma-Aldrich). The IFNα units were calculated for the total transfection volume of 1.2ml to be at IFNα unit concentration of 6U/µl. The plates were returned to the incubator until transfection. Plasmid solution was prepared by adding 8 µl of ADAR p110 plasmid, 8 µl of 5′ UTR model plasmid, both at concentrations of 100ng/ul, to 84 µl of

OptiMEM. 100 µl of transfection solution was made by mixing 4 µl Lipofectamine 2000 with 96 µl of OptiMEM per well, and incubating at room temperature for 5 minutes. Following incubation, the transfection solution was mixed with the plasmid solution, and incubated for 15 minutes. After incubation, 200 µl of the prepared transfection reagent was added dropwise to each well containing 1ml of IFNα – containing media, and the plates were returned to the incubator for a 24-hour transfection. On experiment day 3, 24 hours after plasmid transfection, cell media was gently aspirated, and cells were detached with 100 µl of 0.25% Trypsin-EDTA, and mixed with 1.7 ml of media containing IFNα at a total volume unit concentration of 6U/µl.

5.3.5 *RNASEH1* steric block ASO reverse transfection of HeLa cells with RNAiMAX for protein harvesting in 24-well plates

100 µl of transfection solution was prepared by mixing 1.5 µl of RNAiMax in 48.5 µl of OptiMEM and combining it with 50 µl of ON diluted in OptiMEM. After a ~10-minute incubation, 100 µl the transfection solution was added to each well. Finally, 500 µl of HeLa cells in complete media without Pen/Strep, and at a cell density of 1×10^5 per well were added and gently mixed. An ASO ON gapmer targeting *MALAT1* with a sequence (UGCCAGGCTGGTTATGACUC) was used as a transfection positive control. Protein was harvested after a 24-hour incubation.

5.3.6 Quantitative capillary protein analysis

Fresh 1X RIPA lysis buffer was prepared, with protease inhibitors (Roche, Basel, Switzerland). Cells were washed with PBS, followed by scraping and lysis in 100 µl of cooled lysis buffer. Lysates of two equally transfected wells were combined for a total volume of 200 µl. Lysates were collected in 1.5 ml microcentrifuge tubes and were centrifuged for 30 minutes at 4°C in a previously cooled centrifuge. The supernatant was collected and stored at -80°C.

A portion of the supernatant was retained to measure concentration using a BCA protein assay kit (ThermoFisher, Waltham, MA) on a SpectraMax M5 (Molecular Devices, San Jose, CA).

Protein lysates were assayed using a Jess automated capillary Western blotting system. The sample, separation matrix, stacking matrix, primary and secondary antibodies, and reagents were loaded into a custom Jess plate. The following steps are automated by the instrument: (i) The separation and stacking matrices are aspirated into each capillary of a capillary cartridge. (ii) Next, the sample is loaded, and the capillaries are lowered to aspirate running buffer (iii) Voltage is applied to separate the sample by molecular weight (iv) UV light is used to immobilize the proteins to the capillary wall (v) The matrix is then ejected and immunoprobing begins (vi) Incubation with a primary antibody is followed by a secondary HRP conjugate incubation (vii) Chemiluminescent substrate incubation is initiated and recorded by a camera as a series of images over time. The data can be viewed as a traditional blot, but quantitative analysis is available as signal is interpreted as areas under the peaks for each target.

While this instrument significantly cuts down on the complexity and processing of traditional Western blots, significant upfront validation is required to optimize sample input, primary antibody dilutions, and ratios of secondary antibodies. Here we demonstrate an example of assay optimization, performed to select the conditions for running the *RNASEH1* steric block experiments. The antibody used for the RNase H1 protein assay was a rabbit polyclonal (15606-1-AP), while Vinculin was used as an endogenous control (V9131, Sigma-Aldrich, St. Louis, MO). In this experiment the RNase H antibody dilution is at 1:100, three sample input concentrations are tested (0.25, 0.5 and 1μg), the Vinculin endogenous control antibody was tested at four dilutions (1:2500, 1:3000, 1:4000, and 1:5000), and finally the secondary conjugate ratios were tested as 10:1 or 15:1 rabbit to mouse ratios (**Figure 43**).

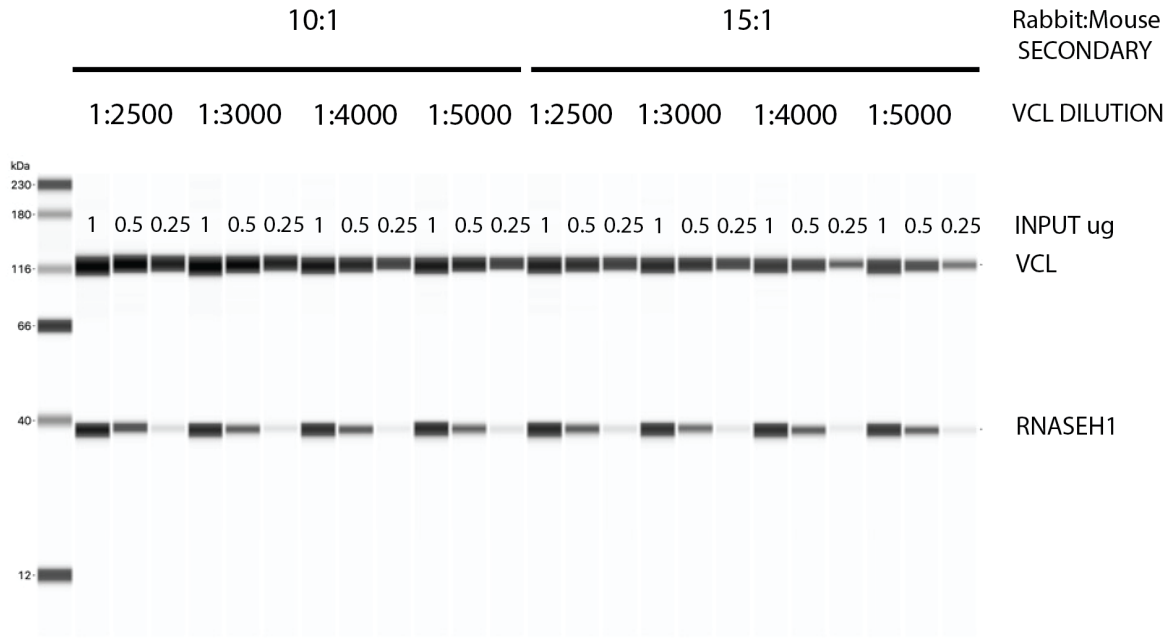


Figure 43. Optimization of Jess conditions for RNase H1 and Vinculin antibodies using HeLa lysate

Automated capillary Western was run on the Jess instrument using RNase H1 rabbit polyclonal and Vinculin mouse monoclonal antibodies. The effects of varying sample input (0.25, 0.5 and 1µg), Vinculin primary antibody dilutions (1:2500, 1:3000, 1:4000, and 1:5000), and secondary conjugate rabbit to mouse ratios (10:1 or 15:1) are shown. Conditions are optimal if both antibodies show similar signal at lower sample input concentrations.

5.3.7 Sanger Sequencing PCR

RNA extraction and cDNA synthesis used as a template for PCR were performed as described in the main Methods section (**2.3 and 2.4**). PCR conditions were optimized for every target separately. Sanger sequencing is not reliable in the first ~50 bases due to technology limitations[395]. This means that if the uORF falls within this region, it cannot be sequenced in the forward direction. A strategy was designed to extend the PCR product at the 5′ by adding a barcode extension to the forward primer, allowing for an optimal positional shift of the uORF. This also allows for all GOI to be sequenced with the same forward primer. The strategy can be further optimized by adding an even longer ~50 nt primer which can allow

for the sequencing of uORFs near the 5′ cap. It should be noted that alternative strategies are required if the uORF is close to the cap to the point where a forward primer cannot be designed. Some targets may be challenging to optimize as there is limited space to place a primer near the cap. This presents a significant obstacle for evaluating editing. Challenging cases may require decapping and ligation steps.

Best efforts were made to place the reverse primer on subsequent exons (i.e. Exon 2 or Exon 3), in order to avoid amplification from genomic DNA, which can have a confounding effect on the interpretation of the RNA editing data. In the event that the uORF was more than 60 nucleotides from the 3′ of the forward priming region, the forward primer was used for sequencing without a barcode extension (**Table 3**). PCR conditions were adjusted based on amplicon size and primer melting temperatures. The T_m Calculator (New England Biolabs, Ipswitch, MA) web application was used to select reaction conditions for the Phusion High-Fidelity DNA Polymerase . Each target required separate PCR condition optimization, which was assessed by automated electrophoresis on a 4200 TapeStation (Agilent Technologies, Santa Clara, CA, USA) (**Table 4**).

PRIMER	DIRECTION	SEQUENCE
RNASEH1_F	Forward	CCGCCTCGGCCTCTGAGGGTGTTGAGCGCCGGCGGCT
RNASEH1_R	Reverse	CGCCTCCGATTCTTGTCCATGT
HOXA11_F	Forward	CCGCCTCGGCCTCTGAGCTACTTCACGGATCCGCTTCAAAGAGGC
HOXA11_R	Reverse	GGGCAGAAAAGAAGGGAGGCTGGA
MTMR14_F	Forward	CCGCCTCGGCCTCTGAGAGGTTCTAGTGTCGGAGTTG
MTMR14_R	Reverse	CAACTTGCTCACCTGTACGG
GAPDH_F	Forward	CCGCCTCGGCCTCTGAGGACACCCACTCCTCCACCTT
GAPDH_R	Reverse	TACATGACAAGGTGCGGCTC
UNIVERSAL_F	Forward	CCGCCTCGGCCTCTGAG

Table 3. Primers used to amplify regions containing editing targets for Sanger sequencing

A list of forward and reverse primers for *RNASEH1*, *HOXA11*, *MTMR14* and *GAPDH* genes. The forward primers use a primer extension (underlined), which allows for preparing Sanger sequencing reaction with the UNIVERSAL_F primer.

The PCR reactions were prepared by mixing 1.25 µl of 10 µM forward and reverse primers, 12.5 µl of 2X Phusion Master Mix (New England Biolabs, Ipswitch, MA), and 5 µl of template cDNA. Exceptions to these conditions are *HOXA11* and *MTMR14* which include 0.75 µl of DMSO. Nuclease-free water was added up to 25 µl final volume. The final reaction and cycling conditions are a result of extensive PCR optimization that minimizes unwanted amplification that may influence Sanger sequencing (not shown). The PCR denaturation, annealing, and extension steps were repeated for 30 cycles.

	GAPDH		RNASEH1		HOXA11		MTMR14	
	Temp °C	Time	Temp °C	Time	Temp °C	Time	Temp °C	Time
Initial Denaturation	98	30 sec	98	30 sec	98	30 sec	98	30 sec
Denaturation	98	5 sec	98	5 sec	98	5 sec	98	10 sec
Annealing	65	30 sec	67	30 sec	65	30 sec	58	30 sec
Extension	72	8 sec	72	8 sec	72	8 sec	72	10 sec
Final Extension	72	5 min	72	5 min	72	5 min	72	5 min
Hold	4	Hold	4	Hold	4	Hold	4	Hold

Table 4. PCR cycling condition for Sanger sequencing

The PCR cycling conditions were optimized for use with the reaction conditions and primer combinations described above. The conditions were optimized to minimize unwanted amplification that may interfere with Sanger sequencing readout.

5.3.8 Automated gel electrophoresis and quality control of uORF PCR

Automated gel electrophoresis was performed to determine the exact amplicon size, concentration, and evaluate any potential contaminant PCR products on a TapeStation (Agilent). PCR contaminant products may have confounding effects on data interpretation and general quality of the Sanger sequencing trace. Therefore, PCR was optimized until contaminant bands represented less than 10% of the overall PCR product. Desired PCR product concentrations were recorded and used to calculate input for Sanger sequencing reactions.

5.3.9 Enzymatic PCR product purification

Amplified PCR products were purified enzymatically in 96-well PCR plates with ExoSAP-IT Express PCR Product Cleanup Reagent (Thermo Fisher Scientific) by adding 2 µl of ExoSAP enzyme to 5 µl of PCR reaction, and incubating on a thermocycler at 37°C for 4 minutes, followed by a 1-minute enzyme inactivation step at 80°C. This clean-up process

removes any excess primers and dNTPs from the PCR reaction and keeps intact double-stranded PCR product without any sample loss. This method was preferred as it is more streamlined than resin-based bead purification systems which require centrifugation and supernatant pipetting of small volumes. The purified intact PCR product can be stored at -20°C or used immediately to setup a Sanger sequencing premix reaction.

5.3.10 Sanger sequencing premix reaction

Samples were prepared for Sanger sequencing, by mixing ~20-50ng of purified PCR product (typically ~2-3 µl of the ExoSAP reaction), with 2.5 µl of either a forward or reverse sequencing primer at a 10 µM stock concentration. The choice of forward or reverse primer was based on the distance of the site of interest from the 5′ of the PCR amplicon, taking into consideration the inherent noise typically observed in the first ~50 bases of Sanger sequencing traces. In most cases, the sequencing primer was one of the primers used for the PCR amplicon generation. Whenever the primer included a barcoded extension at the 5′, a primer specific to the barcode was used to sequence in the forward orientation.

5.3.11 Analysis of Sanger sequencing trace data

Each Sanger trace data file was analyzed manually, using the publicly available editR software, available at the URL: http://baseeditr.com/, with an associated GitHub project URL: https://github.com/MoriarityLab/EditR[396]. Percent G, (or C if the reverse primer was used), was recorded in GraphPad Prism for further analysis. Mock-treated, or control oligonucleotide treated samples were used to establish a baseline at the editing site. The edited site is represented as an overlaid double peak at the editing site (**Figure 44**).

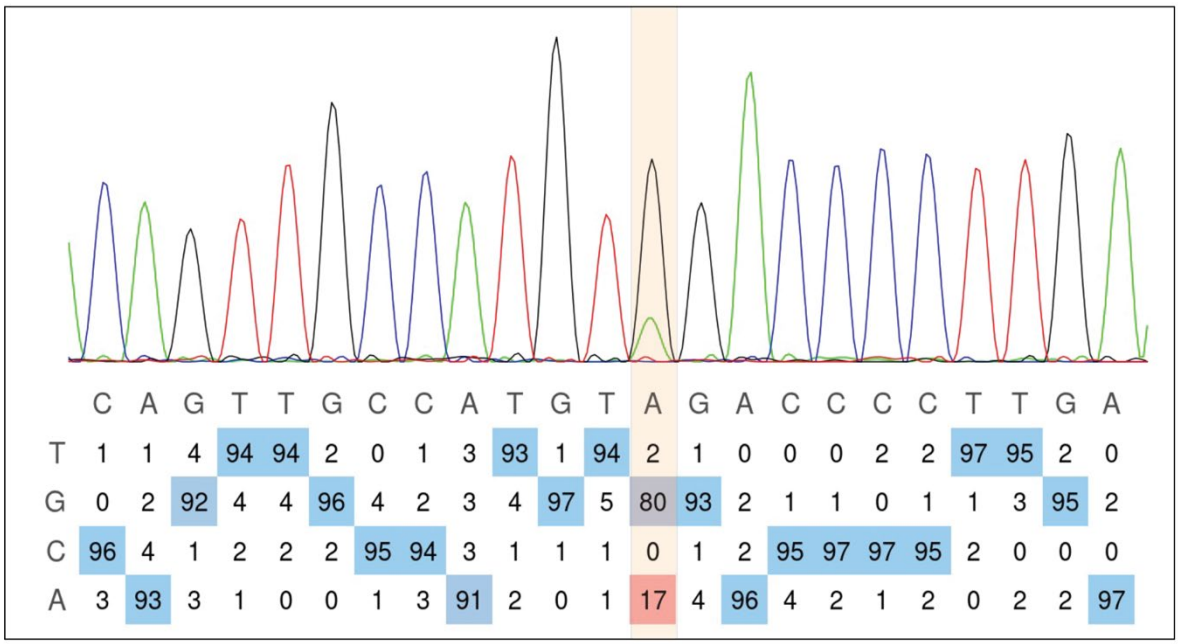


Figure 44. Example of RNA base editing in the 3′ UTR of *GAPDH* analyzed by EditR

Sanger sequencing trace analysis and peak percentage estimation using EditR, demonstrating ~80% of A-to-G conversion, after treatment with ADAR-recruiting ONs.

5.3.12 Formulation of double-stranded (DS) ADAR-recruiting ON designs

Each passenger strand was formulated as a duplex with the guide RNA, by placing a passenger/guide mixture in a water bath heated at 90°C and allowing the water bath to slowly cool to room temperature.

5.4 Results

5.4.1 Targeting uORFs with steric block ONs

5.4.1.1 Steric block ASOs targeting uORF initiation codons are not active promoters of upregulation

Recent work by Ionis Pharmaceuticals described the ability to upregulate expression at the pORF by targeting uORF initiation codons with steric block ASOs. While the concept of applying steric block oligos at the 5′ UTR is unusual as it may interfere with translating ribosomes, the results were both surprising and puzzling. Specifically, Liang *et al.* observed a dose-dependent upregulation with a bell curve shaped dose-response, with the effect of upregulation being lost at higher concentration. While a clear mechanism of action for the observed upregulation effect has not been established, the described ONs utilized accessible designs and chemistry that could potentially be either be improved or used as controls for subsequent experiments[216].

The work presented in Chapter 2 revealed numerous uORFs that are more repressive in comparison to that of the *RNASEH1* gene. To better understand the derepression potential of steric block ONs that target uORFs, ONs were designed to target the uORF of the *HOXA11* gene. The ON designs utilized the same chemistry as described in Liang *et al*[216]. ONs targeting the *RNASEH1* uORF were designed to serve as controls, some of which are identical in both sequence and chemistry as the ones shown to be active by Liang *et al*[216].

ONs were first tested by cotransfection with the pDLR HOXA11 5′ UTR and pDLR *RNASEH1* 5′ UTR wild-type vectors in HEK293T cells. The wild type version of the vectors contains a full length 5′ UTR of the *HOXA11* or *RNASEH1* genes, which both contain a single uORF. ONs were designed at 3 different lengths (14, 16, and 18 nt) using full 2′-OMe PO, 2′-

OMe PS, 2′-MOE PO, 2′- MOE PS chemistries (**Figure 45a**). A non-targeting 18 nt control was designed with all four chemistries. The ONs were designed to cover the initiation codon of the uORFs with the first three bases of their 5′ ends, with the larger portion of the ONs being complementary to the sequence preceding the uORF.

As previously observed, there is a clear derepression of translation at the pORF when the uORF initiation codons of both *HOXA11* and *RNASEH1* get modified from AUG (WT) to UUG (MU). A scrambled non-targeting control 18 nt ON with 2′-OMe PO, 2′-OMe PS, 2′-MOE PO, 2′- MOE PS chemistry surprisingly showed a dose-dependent upregulation effect, while dosing with uORF-targeting ON versions of varying lengths (14,16, and 18nt) resulted in a dose-dependent inhibition rather than upregulation (**Figure 45b,c**). These data were normalized to a non-targeting 18 nt control with the same ON chemistry. However, given that there was a sequence-specific upregulation effect observed with this chemistry, the data was also normalized against the mock treated wild-type vector **(Figure 46a,b)**. At the 40nM dose, where the ON control was not significantly different from the mock treated WT vector, no significant upregulation was observed for *HOXA11* or *RNASEH1*, but some chemistry-dependent effects were evident with the ONs containing 2′-MOE-modified nucleotides.

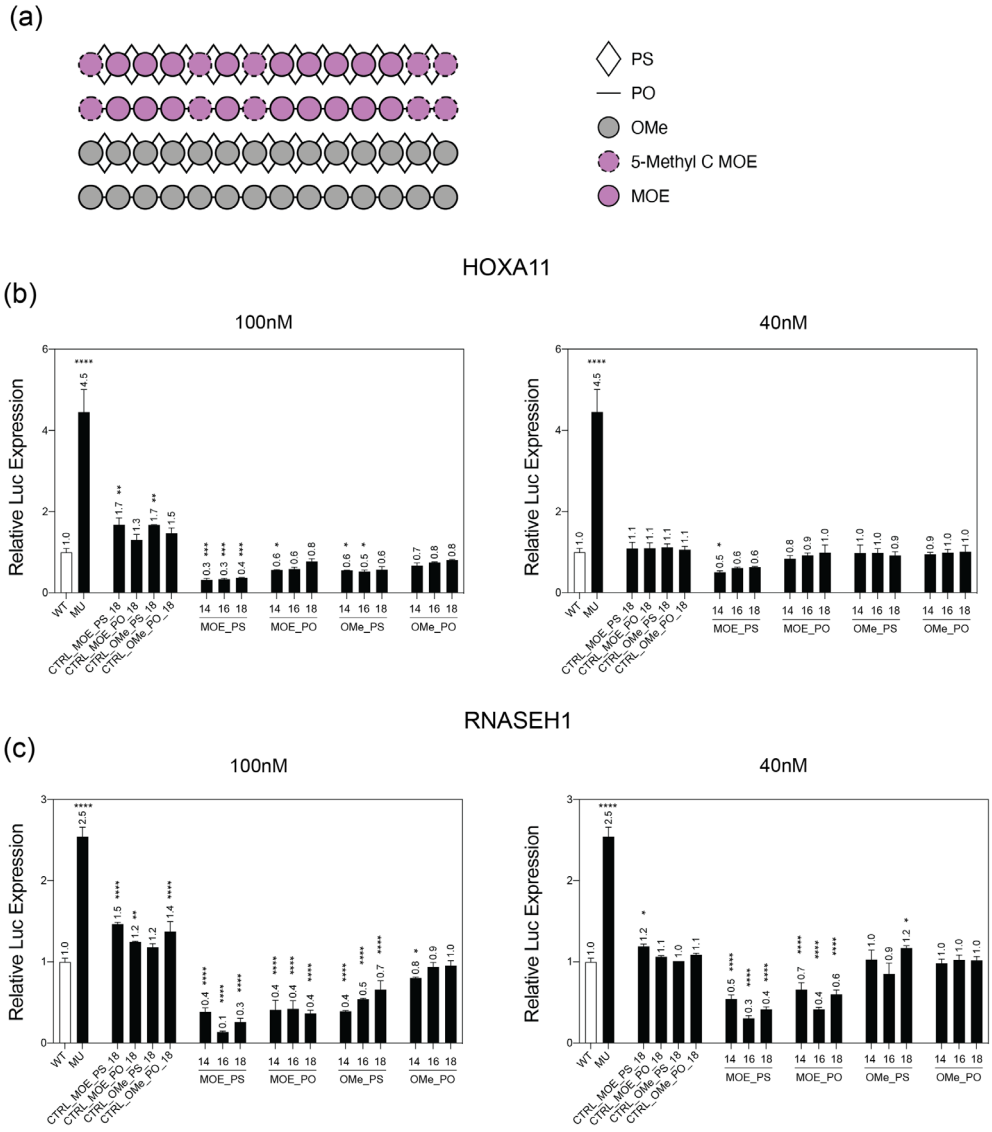


Figure 45. uORF targeting ONs downregulate expression at the RLuc pORF (relative to control ASO treatment)

A cotransfection of plasmid and ON results in a dose-dependent downregulation of RLuc pORF translation normalized to corresponding chemistry non-targeting control ASO (a) Design and chemistry of oligonucleotide uORF-targeting steric blocks. The circles represent nucleotides, the circle colors represent chemical modifications, while the circle connectors depict backbone modifications. (b) Effects of *HOXA11* uORF-targeting ASOs on relative luciferase levels at 100nM and 40nM concentrations. (c) Effects of *RNASEH1* uORF-targeting ASOs on relative luciferase levels at 100nM and 40nM concentrations. Error bars represent ±SD of ≥3 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the mock-treated pDLR GOI 5′ UTR wild-type (*$P \leq 0.05$, **$P \leq 0.01$, ****$P \leq 0.0001$). ASOs were normalized to a non-targeting negative control with the same chemistry.

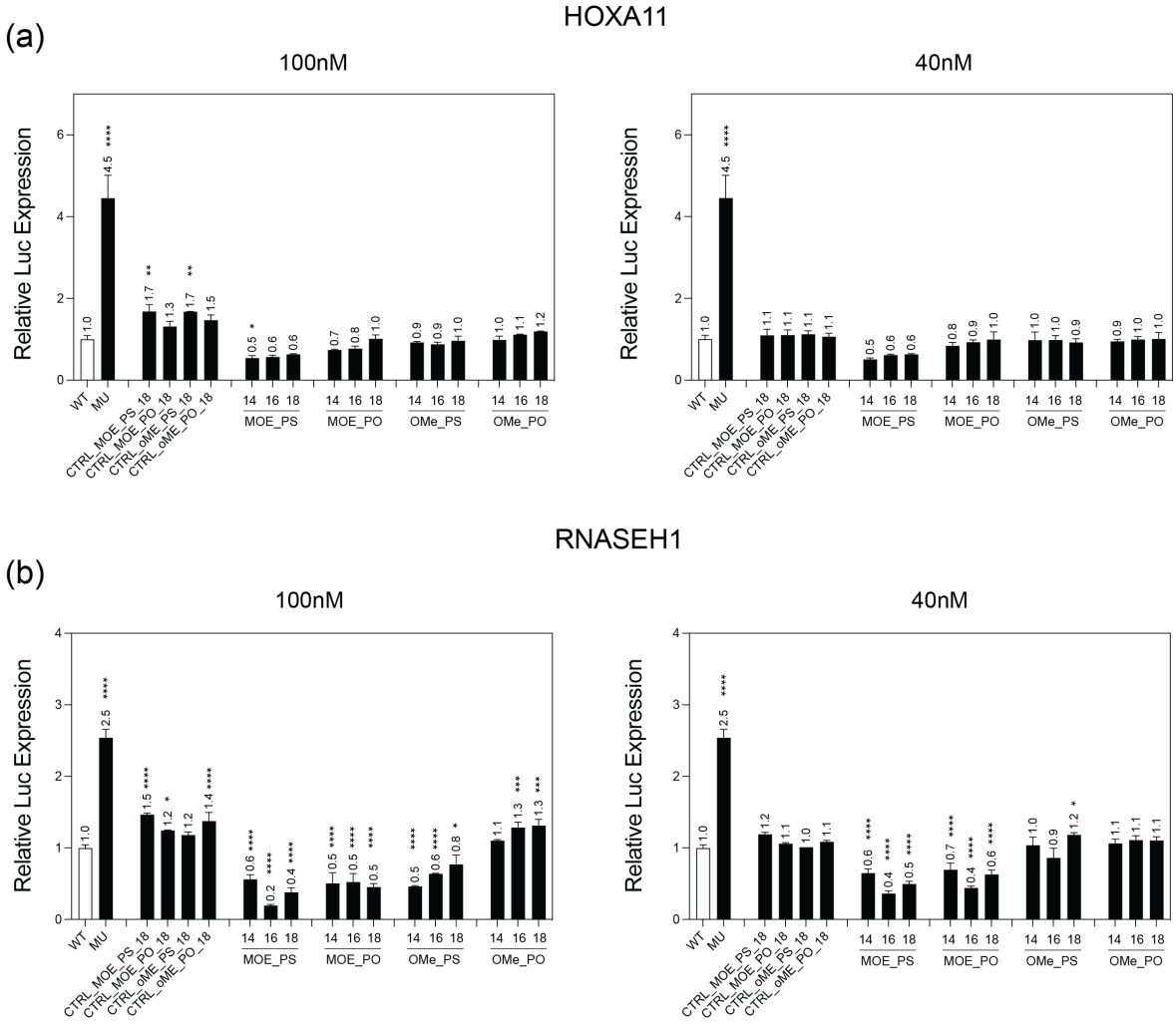


Figure 46. uORF targeting ONs downregulate expression at the RLuc pORF (relative to mock treated wild type vector)

A cotransfection of plasmid and ON results in a dose-dependent downregulation of RLuc pORF translation normalized to mock treated wild-type plasmid ASO (a) Design and chemistry of oligonucleotide uORF-targeting steric blocks (b) Effects of *HOXA11* uORF-targeting ASOs on relative luciferase levels at 100nM and 40nM concentrations. (c) Effects of *RNASEH1* uORF-targeting ASOs on relative luciferase levels at 100nM and 40nM concentrations. Error bars represent $\pm$SD of ≥ 3 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the mock-treated pDLR GOI 5′ UTR wild-type (*$P \leq 0.05$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

To eliminate the possibility of the ONs interfering with transcription or transfection, a two-step transfection strategy was attempted as described by Liang *et al*[216]. First cells were

transfected with plasmid DNA with Lipofectamine 3000. The following day cells were collected and transfected with ONs using RNAiMAX. A dose response experiment was performed testing a range of doses from 1-100 nM for two 16 nt, 2′-OMe, phosphodiester molecules. The *RNASEH1* uORF-targeting 2′-OMe phosphodiester molecule was identical to the one previously described in literature[216]. No significant change in luciferase expression was observed when targeting the uORF loci in the cloned 5′ UTRs of the *RNASEH1* or *HOXA11* genes (**Figure 47**).

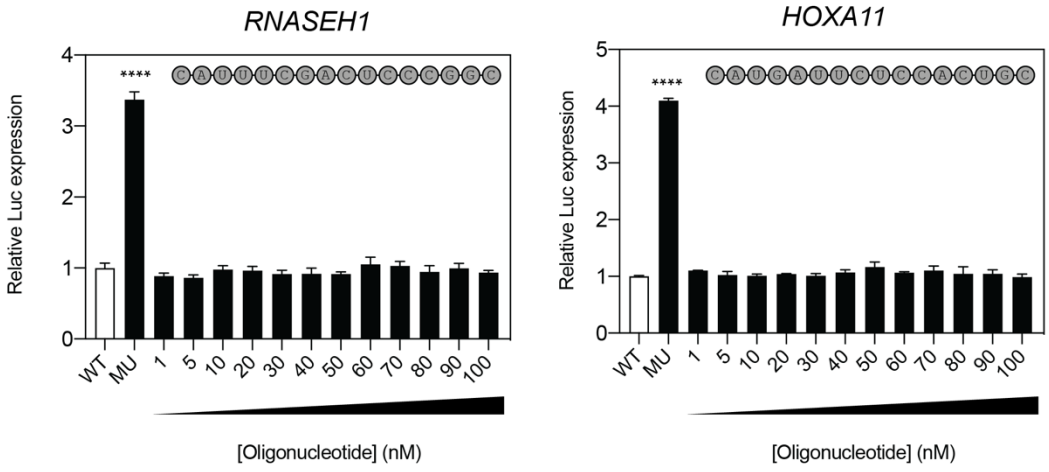


Figure 47. Dual transfection plasmid and 2' O-Methyl modified ON targeting the uORFs of *RNASEH1* and *HOXA11*.

Relative luciferase expression of ON treatment is shown normalized to the mock treated WT pDLR *RNASEH1* 5′ UTR and pDLR *HOXA11* 5′ UTR (white bar). Plasmid transfection was performed with Lipofectamine 3000, while ON was transfected as a secondary transfection using RNAiMAX. Error bars represent ±SD of ≥3 biological replicates. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. No significant change was observed.

Finally, to remove the potential complexity that may be introduced from using a plasmid DNA vector, ONs were directly transfected using RNAiMAX, and endogenous upregulation was evaluated at a protein level using an automated capillary Western technique. No significant upregulation was observed with either PO or PS 2′-OMe chemistries (**Figure**

48). Given that this experiment lacked a transfection control, an ASO targeting *MALAT1* was used, and confirmed >80% knockdown at the 20nM dose (not shown). This experiment utilized a published ON sequence with identical chemistry, which was transfected in HeLa cells using the same transfection protocol. Despite replicating the published conditions, we observed that the steric block uORF-targeting approach was not readily reproducible.

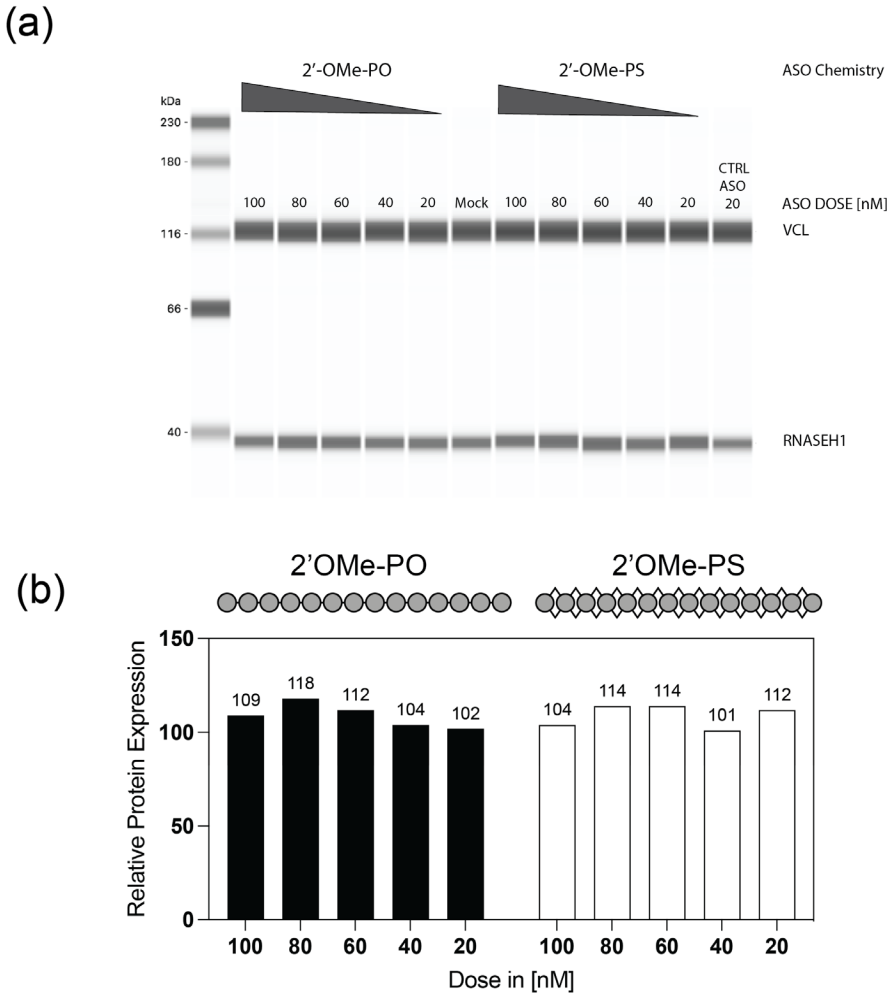


Figure 48. RNase H1 protein levels after treatment with steric block ONs targeting the uORF of *RNASEH1* in HeLa cells

RNase H1 protein levels were assayed on the Jess automated Western platform after a 24-hour dose-response treatment with 2′-OMe ONs with a PO or PS backbone modification. (a) Automated Western blot of signal measured for RNase H1 and Vinculin (VNC) (endogenous control). (b) Signal areas were

normalized relative to a mock treated sample for 2′-OMe ONs with a PO (black bars) or PS backbone modification (white bars).

5.4.2 Targeting uORFs with RNA editing ONs

Advancements in the field of genome editing have opened up new possibilities for targeting disease[376]. While the final frontier of these technologies is a functional cure for many diseases, the complexity of the systems, and potential permanent off-target editing, necessitates significant development before their rapid adoption is realized. It was recently demonstrated that targeted editing is not exclusive to DNA, and that ONs can be used to recruit the ADAR class of deaminase enzymes to elicit A-to-I RNA editing[288,289]. RNA editing may be an effective and potentially safer alternative to genome editing, due to the tunable and transient nature of treatment with ONs. A potential disadvantage of ON treatment is the requirement of repeat dosing.

First generation RNA editing systems, such as CRISPR-Cas13, engineered ADAR-ON, or RNA binding peptide-ADAR fusions, rely on the introduction of a targeting molecule (ON, peptide) and exogenous enzyme[397-401]. However, the delivery and expression of exogenous enzyme has been associated with off-target editing, which represents a risk for drug development[402,403]. Recently, several structurally diverse RNA editing ON triggers have been described; all focused on recruiting endogenous ADAR for site-directed editing. They include RESTORE (Recruiting Endogenous ADAR to Specific Transcripts for Oligonucleotide-mediated RNA editing), LEAPER (Leveraging endogenous ADAR for programmable editing of RNA), chemically modified A-to-I oligonucleotides (AIMer), ADAR RNAs (adRNAs), circular ADAR RNAs (cadRNAs), and Axiomer RNA editing[298,299,404,405].

The mammalian ADAR family of proteins consists of three genes: *ADAR1*, *ADAR2*, and *ADAR3*[290]. The ADAR1 protein has a shorter, constitutively expressed, predominantly nuclear isoform (p110), and a longer interferon-induced isoform, cytoplasmic isoform (p150) [291,292]. ADAR1 and ADAR2 are catalytically active and are recruited by double-stranded RNA substrates. ADAR3 which is catalytically inactive, is expressed almost exclusively in the brain, where it is thought to act as a competitive inhibitor of ADAR1 and ADAR2[406].

All ADAR-recruiting technologies must either introduce or create a double-stranded RNA (dsRNA) substrate near the editing site. To achieve this, some technologies have explored a diverse set of architectures, including introducing a stem-loop universal ADAR recruiting domain (RESTORE), or introducing multiple stem-loops in open or circular architectures (adRNAs, and cadRNAs)[299,403,404]. An alternative approach is to create a dsRNA by complementarity to the mRNA (Axiomer, LEAPER, AIMer)[298,300]. Except for the AIMer platform, all other dsRNA triggers have significant sequence lengths of ~70 nt. In the context of drug discovery, long ONs may be difficult to synthesize in chemically modified forms and would require specialized delivery solutions such as viral delivery or the use of lipid nanoparticles[298]. It was recently shown that long dsRNA triggers may not be required for the recruitment of ADAR, as 30 nt ADAR-recruiting ONs were shown to have optimal activity *in vivo*[298].

The work presented in this Chapter evaluates the utility of A-to-I RNA editing technologies in silencing uORFs, by modifying uORF initiation codons from AUG-to-IUG (interpreted as GUG)[407]. While GUG has been described as a possible initiation codon, its efficiency has been estimated to be low in comparison to that of the canonical AUG initiation codon[82,213,334]. Silencing of uORFs may be of significant therapeutic benefit in multiple disease conditions where upregulation of the pORF translation is desirable.

Given that the endogenous RNA editing ON triggers were all published in the last two years, many scientific questions remain to be answered. The work presented here focuses on demonstrating the ability to edit an AUG codon, and the ability to target an AUG codon in the 5′ UTR both of which have not been attempted before with RNA editing ONs. For this purpose three different ON types were used: (i) Stem-loop designs inspired by the RESTORE technology described in Merkle *et al.*[299] (ii) Novel double-stranded designs that arose from the difficulties to synthesize chemically modified long stem-loop ONs, and (iii) AIMer designs using chemistry described in Monian *et al.* [298](**Figure 49**)

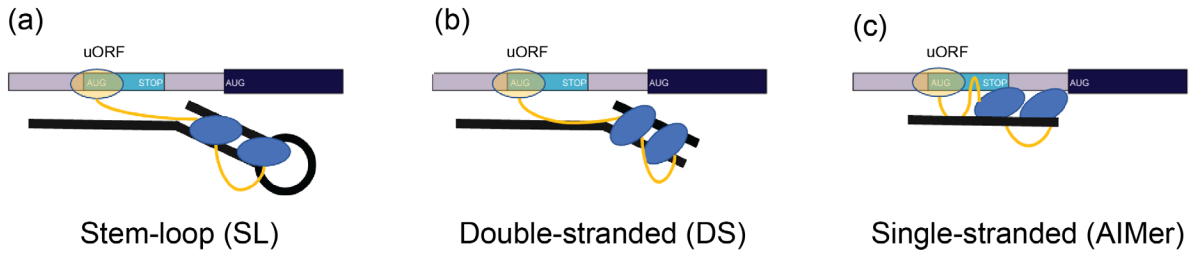


Figure 49. Oligonucleotide designs used to target uORF initiation codons

A schematic of three ON designs used in this study to evaluate RNA editing of uORF initiation codons. (a) RESTORE-inspired stem-loop designs (SL) (b) Double-stranded designs and (c) single-stranded AIMer designs.

5.4.2.1 RNA editing with double-stranded (DS) ONs

The synthesis of ~70 nt chemically modified ONs was both challenging and expensive. One strategy to try to reduce cost was to remove the loop and split the stem-loop design into two ONs (**Figure 49b**). With this design there is a single guide RNA with a specificity domain targeting the *HOXA11* uORF initiation codon, and a variable dsRNA domain based on a shorter passenger strand. This allowed for the testing of different passenger chemistries with a single guide strand (**Figure 50a**). The designs were formulated as a duplex, and were tested in a dual

transfection method, where plasmids expressing either ADAR1 p110 or ADAR p150 were cotransfected with the pDLR *HOXA11* 5′ UTR, and the ONs were transfected the following day with RNAiMAX. It was initially observed that the 2′-MOE and 2′-OMe chemistry passenger strands promote upregulation (**Figure 50b**). While this appeared to be due to RNA editing of the *HOXA11* initiation codon, subsequent experiments using the 2′-MOE passenger sequences and a non-targeting RNA Guide sequence also showed upregulation; suggesting that the upregulation was due to a chemistry-dependent effect in the dual luciferase assay, rather than RNA editing (**Figure 51**).

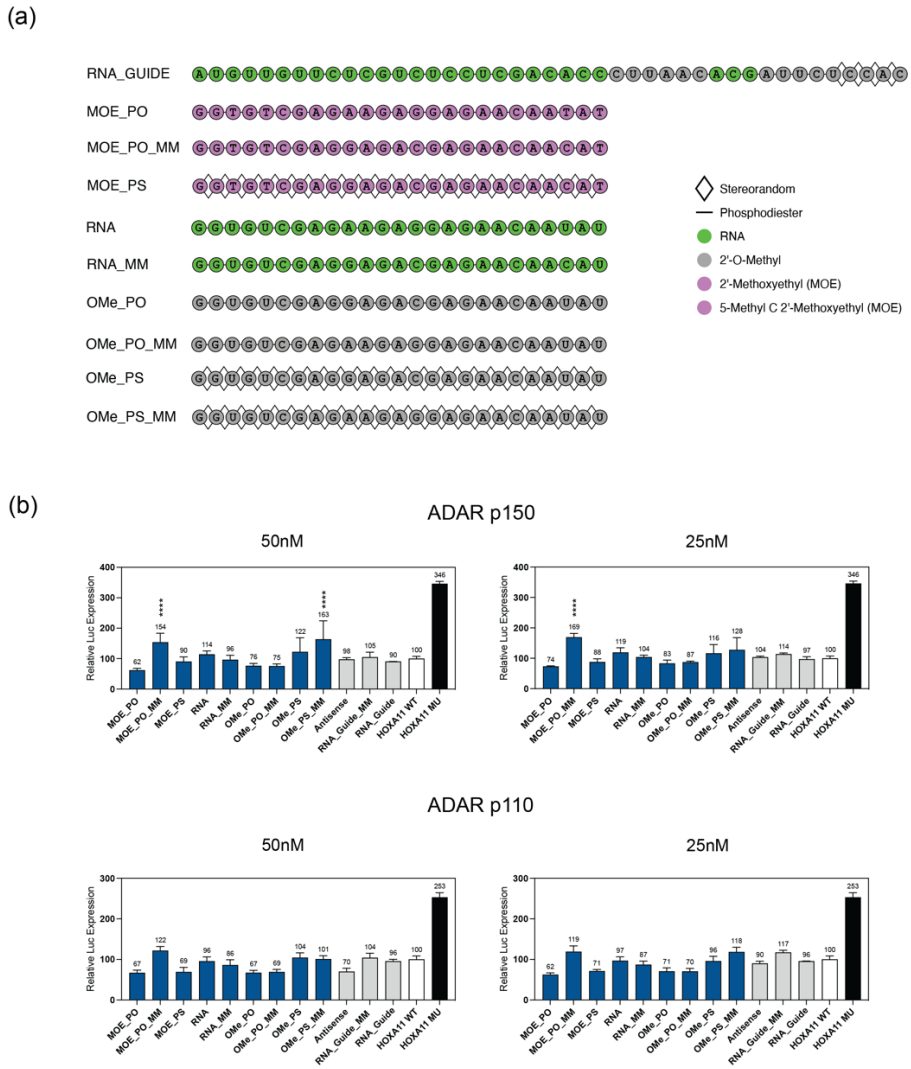


Figure 50. Transfection of double-stranded (DS) RNA editing ON designs

(a) Double-stranded (DS) RNA editing ONs were designed as an asymmetric duplex of a long guide RNA with a specificity domain, and shorter complementary passenger sequences with different chemistries as in (**Figure 49b**). Where the longer guide strand has a specificity domain cognate to the target uORF initiation editing site and forms a DS ADAR recruiting domain by interacting with a passenger strand. (b) Dual-Glo Luciferase reporter assay experiment with DS ONs transfected in HEK293T cells. The blue bars represent DS ONs which all share the same guide strand (RNA_GUIDE) and are named after the chemistry of the passenger strand. (b) Dual transfection experiment representing relative luciferase levels normalized to the HOXA11 wild type plasmid. ONs were transfected with RNAiMAX in cells previously transfected with pDLR HOXA11 5′ UTR wild type and a plasmid expressing either ADAR p110 or ADAR p150. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the mock-treated pDLR HOXA11 5′ UTR wild-type (****$P \leq 0.0001$).

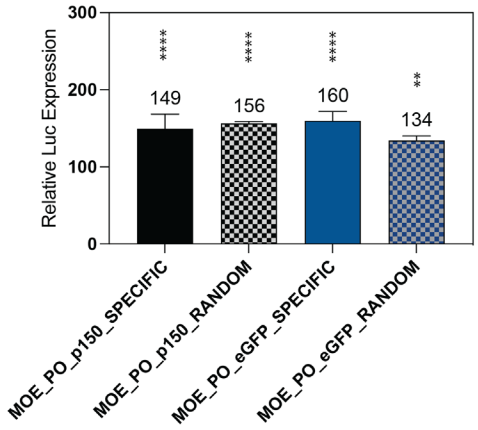


Figure 51. The observed upregulation effect of the 2′-MOE DS ONs is not due to sequence-specific RNA editing

Evaluation of RNA editing of the uORF initiation codon Adenosine in the pDLR *HOXA11* 5′UTR wild type vector with transfected double-stranded RNA editing guide/passenger ON designs. Dual-Glo luciferase assay was performed after 72 hours. The guide strand specificity domain was either randomized to not be complementary to the editing target (RANDOM), or was designed to be cognate to the *HOXA11* uORF region (SPECIFIC). The experiment was performed by cotransfecting ADARp150, or a control eGFP vector. The upregulation effect was recapitulated, but was found to not be either dependent on ADAR p150 or a specific interaction at the editing site. Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to the mock-treated pDLR HOXA11 5′ UTR wild-type with the corresponding ADAR or eFGP plasmid treatment (**$P \leq 0.0001$, ****$P \leq 0.0001$).

Interferon alpha induced ADAR expression in HeLa, SF8628 and HEK293T

The cytoplasmic expression of the longer, catalytically active ADAR1 (p150) can be induced by interferon alpha (IFNα)[291,292]. This property of ADAR was exploited to enhance ADAR activity. The conditions for p150 induction were previously described by Merkle *et al*[408]. The levels of total ADAR1 and ADAR (p150) were measured pre and post induction with IFNα in the HeLa, SF8628 and HEK293 cell lines. Total ADAR expression was doubled in all three cell lines following treatment (**Figure 52a**), while the p150 variant was differentially induced, with HEK293T cells showing significant sensitivity to IFNα treatment

and a 7-fold increase in ADAR p150 expression (**Figure 52b**). ADAR p150 expression was lowest in HeLa cells. IFNα ADAR induction was utilized to maximize the probability of editing a uORF initiation site. While extensive optimization of ON chemistry was beyond the scope of this work, treatment without ADAR induction would be highly desirable for future applications.

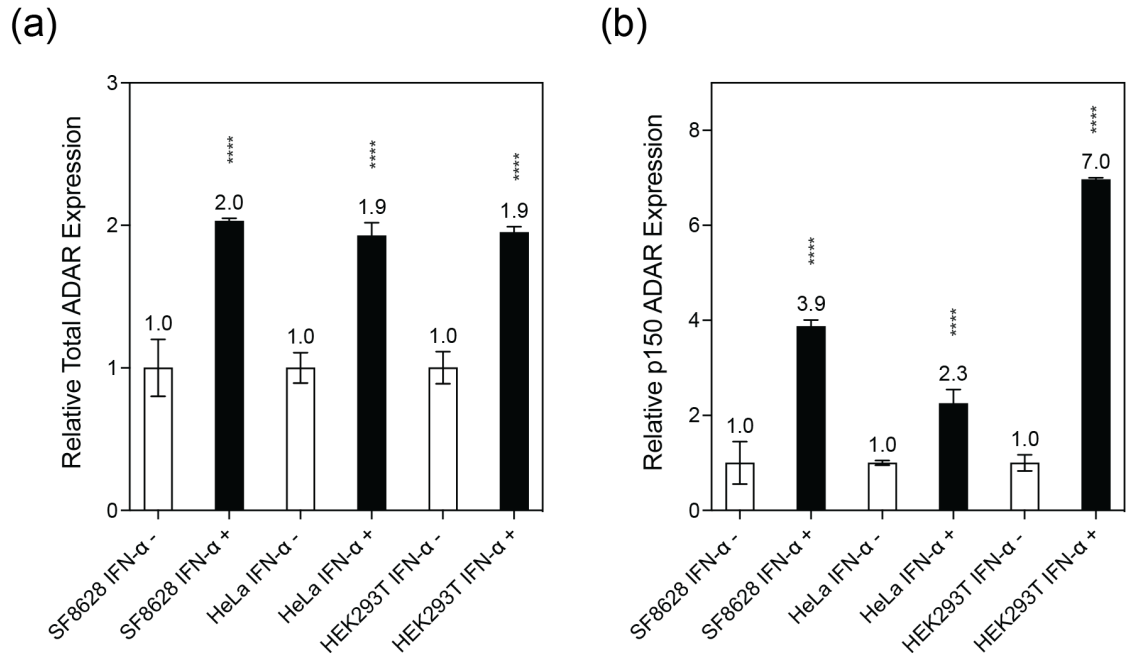


Figure 52. Gene expression analysis of IFNα induced ADAR expression in HeLa, SF8628 and HEK293T cells

Gene expression analysis using qRT-PCR to evaluate the levels of total ADAR1 and ADAR1 p150 in HeLa, SF8628, and HEK293T cells before and after IFNα treatment. (a) Normalized total ADAR expression (black) relative to untreated (white), (b) Normalized ADAR1 p150 expression (black) relative to untreated (white). Statistical significance was evaluated by one-way ANOVA with Bonferroni *post hoc* test. Asterisks indicate significant difference compared to untreated samples in the same cell line (****$P \leq 0.0001$).

5.4.2.2 Endogenous A-to-I RNA editing with stem loop ONs

Stem loop ONs were designed to target the uORFs of *HOXA11*, *RNASEH1*, *MTMR14* and the same *GAPDH* Adenosine targeted in Merkle *et al* was selected to serve as an editing control[299]. These oligos were designed based on the Merkle *et al.* designs 9.4 and 9.5. The ON chemistry, and stem loop structures were designed according to example 9.4. However, at positions 1 and 2 at the 5′ of the ON, 2′-OMe and PS modifications were introduced. It was hypothesized that this would protect the RNA from nuclease degradation, and would not have a negative effect on activity as similar chemistry was used by Merkle *et al.* in their 9.5 designs[299]. The hairpin dsRNA structure is an adopted sequence from a known ADAR2 target site in the GluR2 mRNA. Endogenous RNA editing was performed in the presence of IFNα, and percent editing was measured by Sanger sequencing. Significant editing was observed using the stem loop ONs targeting an adenosine at the 3′ UTR of *GAPDH.* This result is consistent with published data, despite the minor variability in design chemistry. However, the editing of the uORF targets in the 5′ UTR was minimal, and not significant compared to the untreated sample. Given that editing was saturated at 40nM, subsequent screening was done at 20 and 40nM (**Figure 53**).

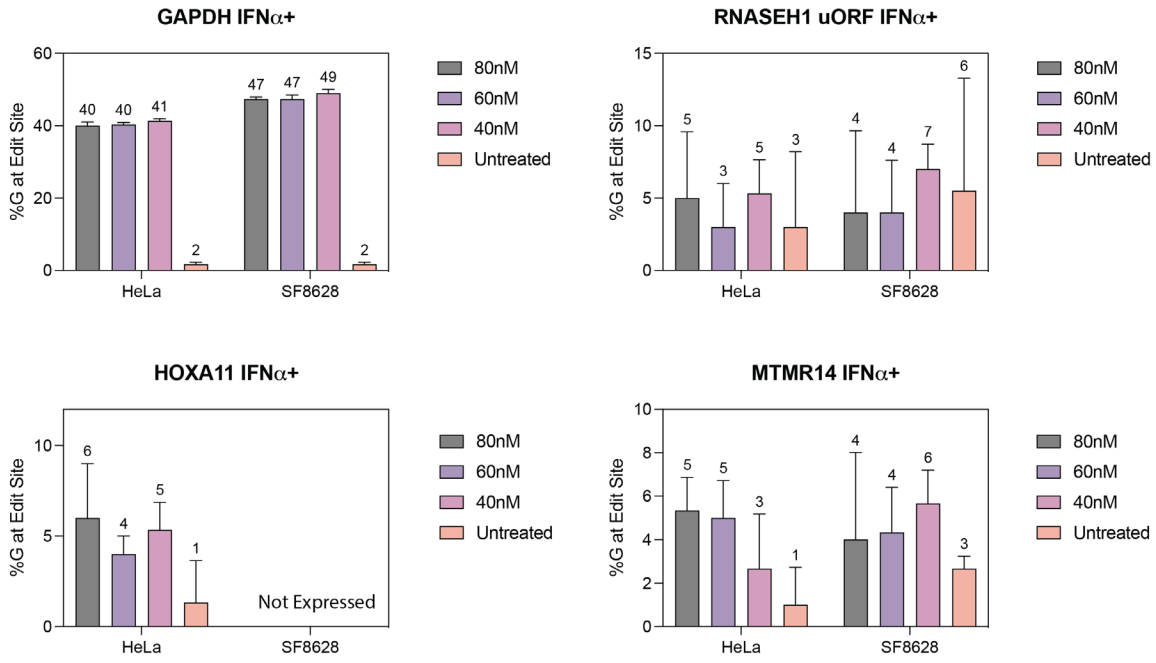


Figure 53. Endogenous A-to-I RNA editing using stem loop (SL) ADAR-recruiting ONs

Endogenous A-to-I RNA editing was performed with SL ADAR-recruiting ONs in the presence of IFNα in HeLa and SF8628 cells and was evaluated by Sanger sequencing. ONs were designed to target an adenosine in the 3′ UTR of the GAPDH gene, and the adenosines in the initiation codons of the *HOXA11*, *RNASEH1*, and *MTMR14* genes. ONs were tested at 3 concentrations 40,60, and 80nM. Analysis of Sanger traces was performed by using the EditR software.

Next, to evaluate the dependance of editing on the levels of ADAR, the SL ONs were tested with or without addition of IFNα. It was observed that editing is significantly more effective in the IFNα induced condition (**Figure 54**).

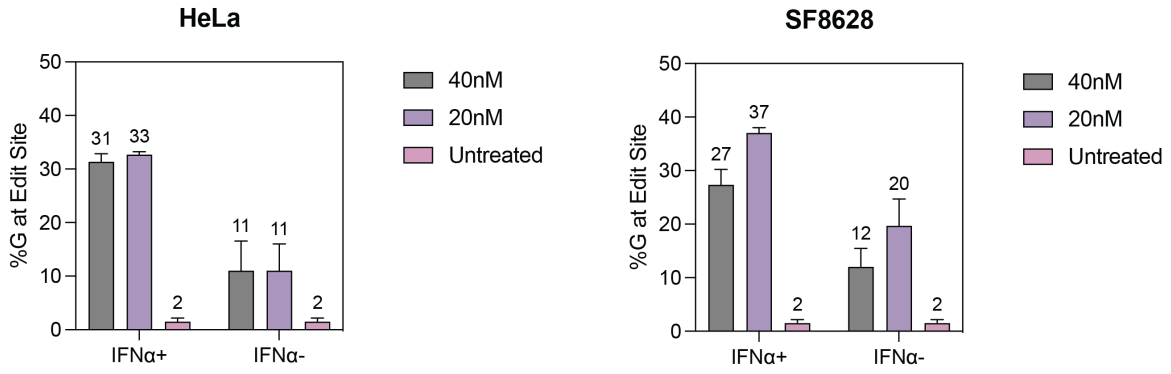


Figure 54. Endogenous A-to-I RNA editing of a target adenosine in the 3′ UTR of the *GAPDH* gene with stem loop ONs is dependent on levels of induced ADAR

Endogenous A-to-I RNA editing was performed with SL ADAR-recruiting ONs targeting an adenosine in the 3′ UTR of the *GAPDH* gene, in HeLa (left) or SF8628 cells(right), and in the presence of absence of IFNα. Percent G at edit site was evaluated by Sanger sequencing. ONs were tested at 2 concentrations 20, and 40nM. Analysis of Sanger traces was performed by using the EditR software.

5.4.2.3 Endogenous A-to-I RNA editing with AIMers

Scientists at Wave Life Sciences recently described a 30 nt chemically modified ADAR recruiting ON molecules (AIMers) with high activity both *in vitro* and *in vivo*[298]. The designs take advantage of high T_m 2′-Fluoro chirally pure PS and PN backbone at the 5′ of the ON, which enables the recruitment of ADAR by creating a ~15-20 bp dsRNA through interaction with the target mRNA (**Figure 55**). This enables the design of significantly shorter ONs, which streamlines ON synthesis, and enables the delivery of ONs without the use of additional delivery vehicles[298]. To test the ability to edit uORFs with AIMer technology, ONs designed against the same targets tested in the SL ON experiments were evaluated for endogenous target editing in the presence or absence of IFNα.

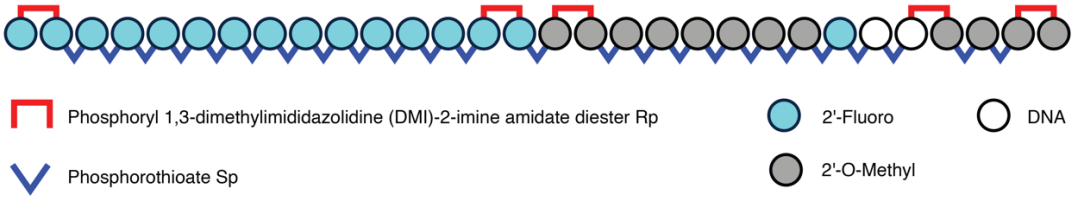


Figure 55. ON chemistry of used AIMers as described by Monian *et al.*

A 30 nt single stranded ON is capable of recruiting ADAR by using 2′-Fluoro chemistry to create Watson-Crick base pairing to the 3′ of the edit site. The edit site is defined as a 2′-Fluoro, DNA, DNA triplet where the center DNA is a mismatched DNA Cytosine, opposite of the target Adenosine. Chiral Rp PN (phosphoryl 1,3-dimethylimididazolidine (DMI)-2-imine amidate diester), and chiral Sp PS(phosphorothioate) backbone is used, with 2′-*O*-methyl modifications in the specificity domain.

Superior editing of the *GAPDH* 3′ UTR target site was observed in both HeLa and SF8628 in the presence of IFNα. IFNα conditions were optimal for editing the *GAPDH* target adenosine. However, ~44% editing at 20nM was observed even without the addition of IFNα-induced ADAR expression (**Figure 56a**). Editing of the uORF initiation codons in *RNASEH1*, *HOXA11* and *MTMR14* was still close to background (**Figure 56b, c, d**). However, 9% of A-to-G conversion was observed in the SF8628 cell line, with the AIMer targeting the uORF of *RNASEH1*(**Figure 56b**). Thus, the critical question to be answered was whether editing can be achieved in the 5′ UTR, as the presence of evolutionary constraints that protect the critical AUG codon from being modified was hypothesized.

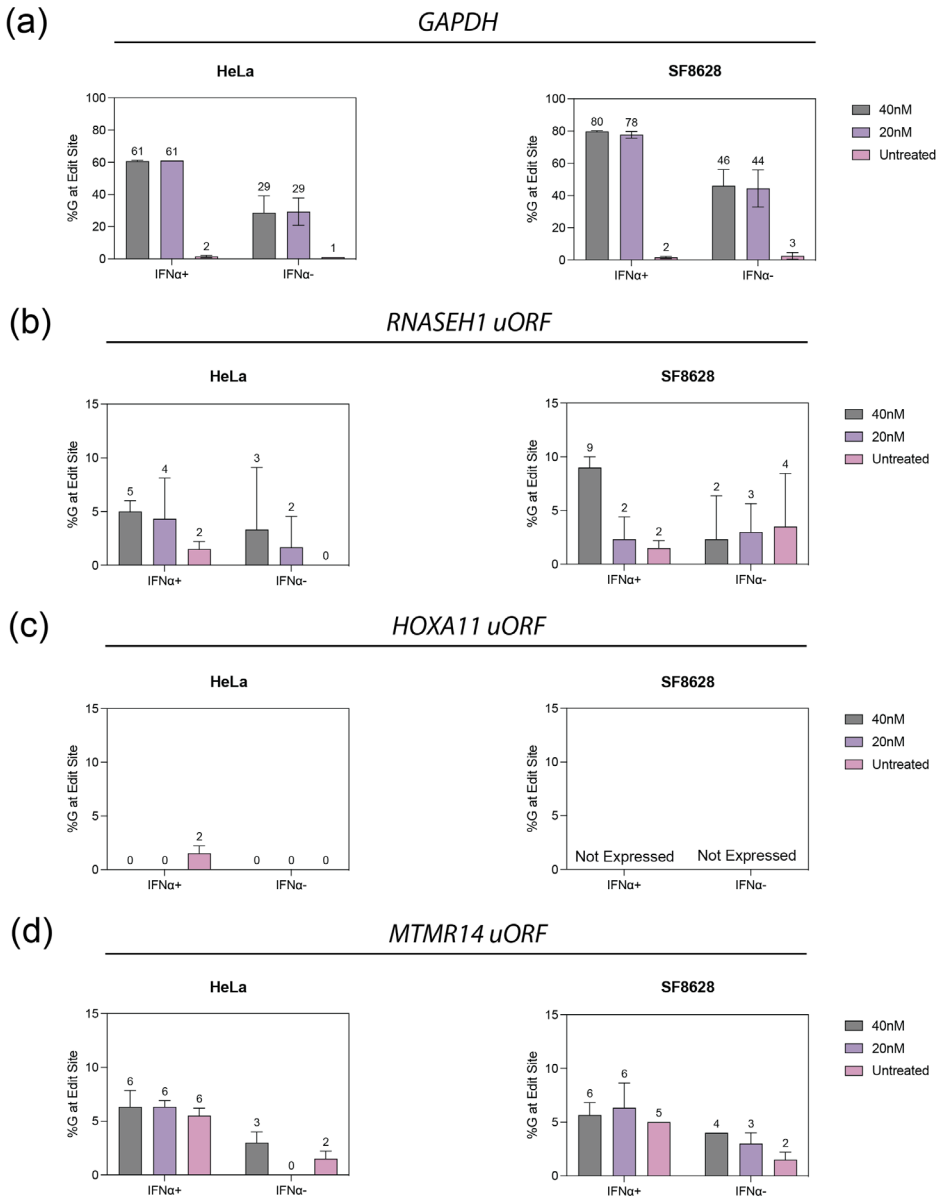


Figure 56. Optimal A-to-I RNA editing with AIMer ONs can be efficacious even without IFNα ADAR induction, while uORF initiation codon editing may be more challenging

Endogenous A-to-I RNA editing was performed with AIMer ADAR-recruiting ONs targeting an adenosine in the (a) 3′ UTR of the *GAPDH* gene, and the adenosines in the initiation codons of the uORFs in (b)*RNASEH1*, (c)*HOXA11*, and (d)*MTMR14* genes. A-to-I editing was evaluated in the presence or absence of IFNα in HeLa (Left) and SF8628 cells (Right). Percent G at edit site was evaluated by Sanger sequencing. ONs were tested at 2 concentrations 20, and 40nM. Analysis of Sanger traces was performed by using the EditR software.

5.4.2.4 DNA Inosine placement opposite of a C nearest neighbor improves editing

The efficiency of editing is highly influenced by nearest neighbor bases, where a U nearest neighbor to the 5′ of the target A is the most optimal for efficient editing, a G to the 3′ of the target A is the most optimal 3′ nearest neighbor, and C opposite the target A is the optimal mismatch for editing[405]. While these observations have been studied in context of the LEAPER system, it was taken under consideration that a non-U nearest neighbor at the 5′ may result in poor editing outcomes. Hence, it was hypothesized that the poor editing outcomes of the HOXA11 and MTMR14 genes were likely due to nearest-neighbor effects on editing efficiency.

A possible solution was proposed in a patent by ProQR discussing the Axiomer technology chemistry, where it was mentioned that the 5′-CAA-3′ target sequence paired to a 3′-GCU-5′ with the A-C mismatch at the editing site is disfavored due to a steric clash of the guanosine with an amino acid side chain of ADAR2[409]. It was therefore suggested that a smaller nucleobase, such as I, can form a base pair with C, and potentially fit better without causing a steric clash. Thus, a DNA Inosine base was applied to the ONs targeting *HOXA11* and *MTMR14*, both of which have a 5′ C nearest neighbor base next to the target A.

In addition to this, a secondary AUG site in the GAPDH gene (Site 2) was selected in proximity to the successfully edited Adenosine (Site 1), to assess if an AUG codon can be edited in the 3′ UTR (**Figure 57a**). Given that this site also has a C nearest neighbor, ONs were designed to pair the C with either a G or I.

Surprisingly, despite the very close proximity to Site 1 which can be efficiently edited, Site 2 editing was low. The inclusion of Inosine opposite the Cytosine nearest neighbor improved editing to ~6%, but this was still ~10-fold less than the editing of Site 1 (**Figure 57b**). However, the introduction of Inosine opposite of the Cytosine nearest neighbor significantly improved the editing of the uORF initiation codons in *HOXA11* and *MTMR14*

(**Figure 57c**). This data represents the first endogenous A-to-I RNA editing of a uORF initiation codon in the 5′ UTR, demonstrating that uORF initiation codons can be edited with ON technologies.

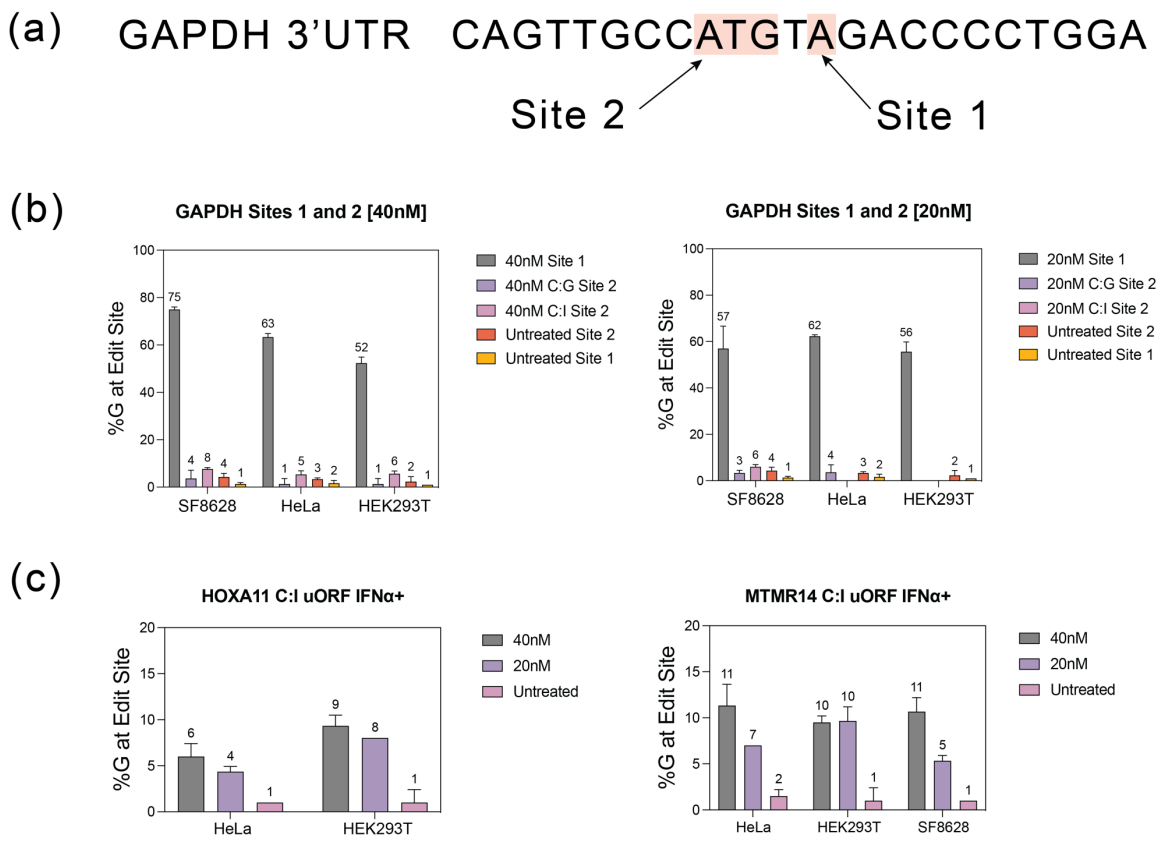


Figure 57. uORF initiation codons can be edited with ON technologies

Endogenous A-to-I RNA editing was performed with AIMer ADAR-recruiting ONs targeting an adenosine in the (a, b) 3′ UTR of the *GAPDH* gene (Sites 1 and 2), and the adenosines in the initiation codons of the uORFs in (c)*HOXA11* and *MTMR14* genes. A-to-I editing was evaluated in the presence of IFNα in HeLa, HEK293T, and SF8628 cells. Percent G at edit site was evaluated by Sanger sequencing. ONs were tested at 20, and 40nM concentrations. Analysis of Sanger traces was performed by using the EditR software.

5.4.2.5 A-to-I RNA editing of uORF initiation codons for the purpose of protein upregulation

Encouraged by the ability to edit uORFs, a new panel of genes was selected to evaluate an AUG to GUG silencing using the pDLR GOI 5′ UTR vector, in a strategy similar to the one described in Chapter 2. The panel was selected using uORF Detective and the uORFeome databases, based on the ribosome profiling initiations signal at the uORF, relative to that of the pORF. This panel includes the genes: *ADAM19*, *ACAP2*, *NOL6A*, *METTL14*, *ARHGAP18*, *AKAP10*, *DEPDC1*, *FIS1*, *ANTXR1*, *KDM5A*, *EDEM3* and *YEATS4*. It can be concluded that an A-to-G mutation silences uORFs, as significant upregulation was observed in most genes when the uORF initiation codon was modified from AUG to GUG (**Figure 58**). This opens the possibility of applying A-to-I RNA editing technologies to target uORFs.

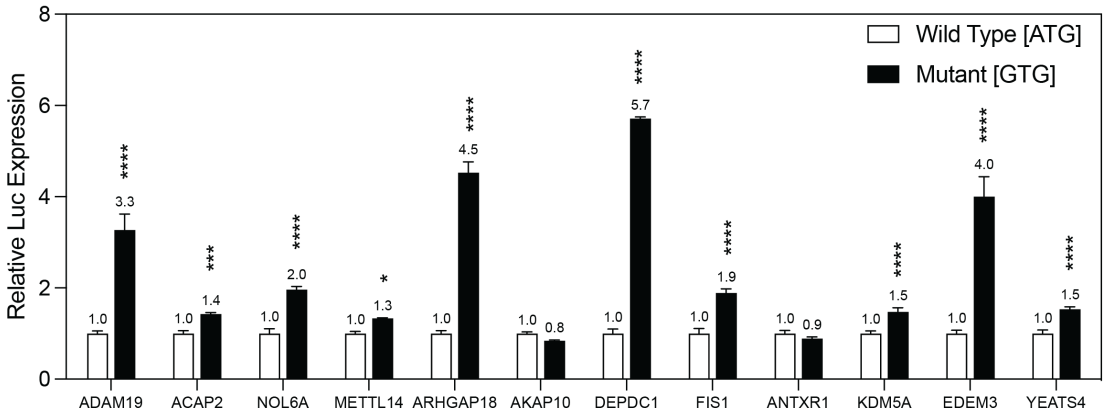


Figure 58. A-to-G mutations in uORF initiation codons silence uORF activity

Relative luciferase expression is shown for reporter constructs in which *bona fide* wild-type (white) or modified (black) 5′ UTRs were cloned before the RLuc initiation codon. The wild-type uORF AUG initiation codons were modified to GUG by site-directed mutagenesis. The RLuc expression of each construct is reported as normalized to the wild-type parent 5′ UTR. Statistical significance was evaluated by a Student's *t*-test relative to the corresponding wild-type control. Asterisks indicate significant difference compared to the mock-treated pDLR HOXA11 5′ UTR wild-type (*$P \leq 0.05$, **$P \leq 0.01$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

Finally, the wild-type plasmids of this gene panel were co-transfected with a vector expressing the ADAR p110 enzyme. The following day the cells were transfected with corresponding AIMers in the presence of IFNα. Significant upregulation of pORF expression was observed in *DEPDC1*, *ANTXR1*, and *EDEM3* genes (**Figure 59**). However, in some genes like *METTL14*, *FIS1*, *YEATS4* and *HOXA11* the pORF expression was downregulated. This may suggest that while there may A-to-I editing, in some cases there may be a steric block effect that inhibits translation. Endogenous editing of the upregulated genes is currently being evaluated.

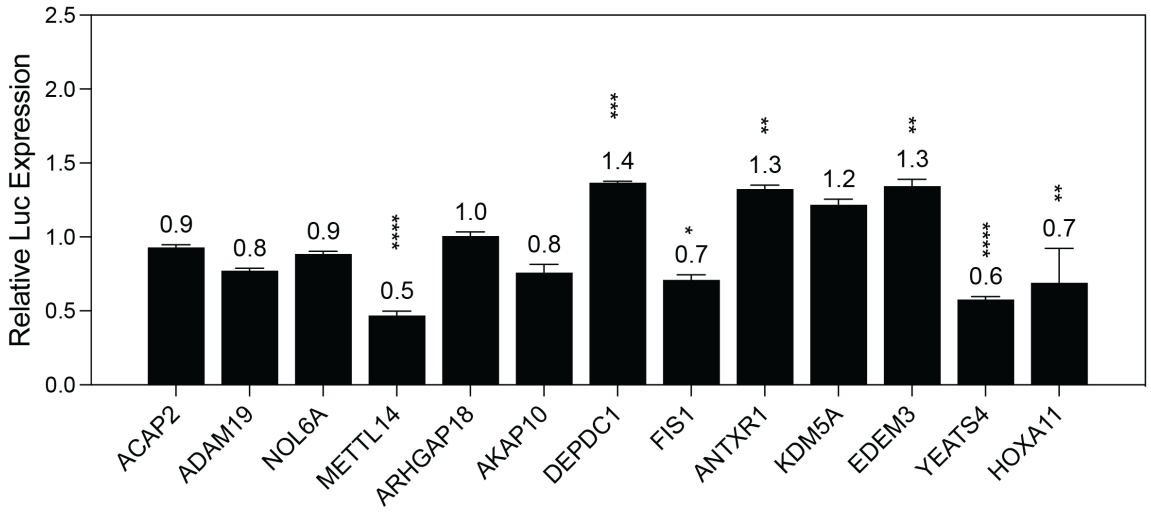


Figure 59. Dual transfection of constructs carrying *bona fide* 5′ UTRs and AIMers in the presence of IFNα show both upregulation and downregulation of RLuc pORF translation

Relative luciferase expression is shown for reporter constructs in which *bona fide* wild-type 5′ UTRs were cloned before the RLuc initiation codon, treated with a A-to-I editing ONs. First plasmid DNA and pDLR GOI 5′ UTR wild-type constructs were transfected in HEK293T cells. The following day, cells were harvested and transfected with ONs. The RLuc expression of each construct is reported as normalized to a treatment control (wild-type parent 5′ UTR treated with an AIMer targeting *ACTB*). Statistical significance was evaluated by a Student's *t*-test relative to the treatment control. Asterisks indicate significant difference compared to the mock-treated pDLR HOXA11 5′ UTR wild-type (*$P \leq 0.05$, **$P \leq 0.01$, ***$P \leq 0.001$, ****$P \leq 0.0001$).

5.5 Discussion

5.5.1 The problem with steric block ASOs in the 5′ UTR

Several years ago, it was reported that steric blocking ASOs of lengths as short as 12 nt targeting the initiation codon of the uORF in the *RNASEH1* can result in targeted upregulation of RNase H1 protein[216]. Conceptually, the results of this study conflict with prior knowledge and utility of steric blocking ONs. For instance, the PMO drug radavirsen was a steric block ON, designed to interact with a target sequence in the matrix segment of influenza A, and block translation[378]. The surprising aspect of the reported data is that the observed upregulation effect is not dependent on ON length and is not consistent regarding the influence of ON binding affinity. For example, a 12 nt 2′-MOE-PS molecule was shown to reduce RNase H1 protein levels to 70%, while a 12 nt 2′-OMe-PS molecule increased the relative RNase H1 protein levels to 191%. These two molecules have melting temperatures T_m of 66.1°C and 59.5°C respectively[216]. Moreover, an 18 nt 2′-MOE-PS molecule with a T_m of >90°C was found to upregulate relative RNase H1 to 178%[216]. This highlights a completely opposite trend in the two most utilized chemistries in the ON field. Furthermore, a loss of upregulation effect was observed with increased ON dose. This observation is difficult to explain, but was consistent across different chemistries[216].

The described ONs were relatively short in length (12-20 nt) and include commonly utilized ON chemistries (2′-MOE, 2′-OMe, PS, and PO), and so undertaking a reproduction study using compounds described in this manuscript was relatively straight-forward. Surprisingly, the data was not reproducible by following the published protocols. Ironically, in the plasmid/ON cotransfection experiment the most potent upregulating ON was the scrambled non-targeting control (**Figure 45**). However, to eliminate any doubt that the scaling was performed against an ON with an anomalous effect, the data were also rescaled against the

untreated wild type pDLR *HOXA11* 5′ UTR and pDLR *RNASEH1* 5′ UTR vectors. No upregulation was observed relative to either chemistry-specific scrambled controls, or untreated cultures (**Figure 46**). Finally, in an experiment targeting endogenous *RNASEH1,* using an automated capillary Western system, no significant upregulation was observed following treatment with any of the uORF blocking ONs. Given that we were unable to reproduce the findings of Liang *et al*. this approach was not further pursued[216].

5.5.2 RNA editing of uORFs

A new class of ON molecules is emerging, adding RNA editing to the diverse portfolio of ON platform mechanisms[218,219]. It has now been described that ONs can be used to recruit the ADAR family of enzymes to elicit a change of Adenosine-to-Inosine with specificity. A very diverse set of ON chemistries and architectures have been shown to be able to recruit ADAR, as the critical substrate for recruitment is double stranded RNA. Most recently, it has been shown that highly efficacious ADAR-recruiting molecules can be as short as ~30 nt, unlocking the possibility of treatment paradigms similar to those of approved ASO molecules[298]. While ADAR-recruiting molecules have yet to enter the clinic, several companies have been advancing RNA technologies including Wave Life Sciences, Korro Bio, and ProQR, which notably signed a $1.5 Billion partnership agreement with Eli Lilly for their Axiomer RNA editing platform. While the promise of A-to-I editing has been focused on the disease subset of the over 20,000 G-to-A mutations in humans, the work presented here proposes that this technology can be used to alter initiation codons. Specifically, the focus of the work presented in this chapter was to evaluate if uORF AUG initiation codons can be edited as a means for uORF silencing, and derepression of translation of the pORF.

In the earlier days of the work presented in this thesis, when the stem loop ONs were the only drug-like RNA-editing molecules described in literature, synthesis and cost challenges

limited the amount of ONs that could be tested. However, unlike the steric block ONs, the designed control ONs targeting *GAPDH* were found to be as efficacious as previously published[299]. The application of these ONs to the uORFs of *HOXA11*, *RNASEH1*, and *MTMR14* genes, first revealed that editing of an adenosine in the 5′ UTR may be more challenging than targeting the 3′ UTR. Moreover, with some of the ADAR proteins being ubiquitously expressed, it was hypothesized that evolutionary constraints against editing the critical AUG codon are possible[410]. In addition to this nearest neighbor preferences have recently been discussed in context of RNA editing ONs[405]. An AUG sequence in close proximity of a very active editing site in the *GAPDH* 3′ UTR was selected to attempt AUG editing outside of a 5′ UTR uORF context. With only a few bases apart, editing of the AUG site was limited.

Efficient RNA editing has been shown to be significantly influences by nearest neighbor bases. In particular, a U nearest neighbor to the 5′ of the target A is preferred[405]. The ability to overcome the negative influence of nearest neighbor effects on RNA editing is a topic of active research. It was recently reported that I in the place of G, opposite of a 5′ C nearest neighbor may alleviate steric hindrance[409]. Given that C was the nearest neighbor of the uORF initiation codons in both *HOXA11* and *MTMR14*, two ONs containing DNA I opposite of the C nearest neighbor were designed. While the influence of this modification was subtle, it was enough to increase the editing of the *HOXA11* and *MTMR14* to ~11% relative to ~1% of background noise in the unedited control. This editing efficiency is low in comparison to that achieved in the *GAPDH* active site in the 3′ UTR, but reveals for the first time that uORF initiation codons can be edited, and that additional chemical modification research can likely further improve editing efficacy.

5.5.3 Future prospects of uORF ON research

Understanding the validity of whether uORFs can be targeted with steric block ASOs is of critical importance to the field. Our work tried to recapitulate some of the published work with intent to test additional chemistries, but an upregulation effect was not observed in both reporter assays and endogenous *RNASEH1* targeting experiments. However, even if these ONs are as active as described, the loss of activity at high doses precludes therapeutic development[216]. Steric blocking ONs targeting the 5′ UTR have been described to upregulate SMN2 levels, by members of our laboratory[411]. However, this mechanism of action was determined to be unrelated to the *SMN2* uORF. Furthermore, the work presented in Chapter 2, demonstrates that the *SMN2* uORF is not consequential for translation of the pORF. It is possible that the steric block upregulation effect is due mRNA stabilization, masking of uncharacterized *cis* protein binding sites, or direct ON interaction with RNA-binding proteins.

RNA editing presents an exciting alternative for uORF interference, although the limitations of this approach have been incompletely characterized. It has been discussed in literature and shown in this work that editing is highly dependent on sequence context[405,409]. More work is required to understand the sequence context limitations, and the possibility to overcome these limitations with the introduction of chemical modifications. The ON recruitment of the ADAR family is limited to the deamination of Adenosine. Harnessing enzymes which can enable the targeting of additional base is a likely future direction for the RNA editing field. Endogenous RNA editing in mammals is thought to be predominantly caused by deamination of (A>I), but also (C>U). The cytidine deamination has been attributed to the APOBEC (activation induced deaminase (AID), apolipoprotein B-editing catalytic polypeptide-like), and the cytidine deaminase (CDA) protein, all of which are capable of hydrolytic deamination of C to U[412]. Recently APOBEC3A-Cas13 guide RNA fusions have been utilized for programmable C to U editing[413]. Given that APOBEC1 catalyzes the

endogenous processing of *APOB* mRNA to generate the truncated ApoB48 proteoform, it is likely that APOBEC proteins can be recruited by specific sequence triggers[414-416].

An interesting aspect of research on deaminases is their implication in the viral response. APOBEC-like C-to-U transitions have been found to be greatly represented in the genomes of small DNA viruses, herpesviruses, retroviruses, and coronaviruses, including the SARS-CoV-2 genome[417]. The ability to recruit deaminases to direct specific editing may be of significant therapeutic benefit.

5.5.4 Conclusions

The work presented in this chapter demonstrates that A-to-G editing of uORF initiation codons is indicative of uORF silencing and supports A-to-I RNA editing strategies for interfering with uORF activity. Novel RNA editing technologies were applied to demonstrate the first A-to-I editing of uORF initiation codons. While uORF initiation codon editing was not as efficacious as the editing of targets in the 3′ UTR of the *GAPDH* gene, it was improved by applying a chemical modification of the ON. ONs have only recently been used to elicit targeted adenosine deamination[298,405,408]. Future ON architecture designs and chemical modification research may unlock the potential of RNA editing technologies.

RNA editing can be used in several ways to interfere with uORF function including uORF silencing by editing uORF initiation codons from AUG-to-GUG, uORF introduction by creating AUG codons at AUA, and interfering with stop codons with UAA-to-UGG, UAG-to-UGG or UGA-to-UGG editing. The work described in Chapters 2 and 3, demonstrates that uORF silencing can lead to significant upregulation of translation at the pORF. This notion makes uORF an attractive targeting region for therapeutic development.

The ability of the ribosome to remove high T_m ONs after editing remains to be elucidated. Long RNA substrates and short 5′-Fluoro-modified ONs with high T_m may inhibit

ribosomal function by promoting editing, but also acting as a steric block ON which may interfere with translation. Interestingly, the stem loop ONs may be preferred in this case, as they engage the targeting region only with their specificity domain.

The work presented in this chapter demonstrates the first RNA editing of uORF initiation codons and highlights the potential of using ON technologies for interfering with uORF initiation as a strategy for gene upregulation. The ability to efficiently disrupt uORF initiation codons can have broad therapeutic applicability in conditions where upregulation of the expression of the pORF is desired.

6 Discussion and conclusions

According to recent estimates nearly 10,000 monogenic diseases arise from mutations in approximately 4,000 genes[285,418,419]. The majority of these diseases can be categorized as rare genetic disorders, based on their low prevalence of ~3.9-6.6 in 10,000 subjects[418]. A recent study suggests that the number of rare diseases is severely underestimated, and that "hyper-rare" diseases, defined as affecting < $1/10^8$ individuals, may outnumber the currently known rare diseases[420]. Despite their individual low prevalence, rare genetic disorders afflict ~260-450 million people worldwide; meaning that as a combined category they are relatively common in the population. Rare disorders have been found to disproportionally affect the nervous system, particularly during development. A concerning statistic is that 83% of all rare genetic disorders are paediatric disorders[418]. There is a significant unmet need for the development of therapeutics for rare disorders, as treatment options are estimated to be available for only ~5% of these conditions[285,419,421]. An additional complication is presented by the long diagnostic process, which is estimated to take between six and eight years, while in some cases may take over a decade[422]. According to a European Organization for Rare Diseases (EURORDIS) survey performed in 2017, more than 70% of patients have incapacitating symptoms and have difficulties with performing daily activities[423]. This presents a significant burden for family members, who often assist patients as informal carers[424]. The economic cost of rare diseases in the United States was estimated to reach nearly $1 trillion in 2019; half of which was attributed to direct care including hospital inpatient care and prescription medication, while the other half was attributed to loss of productivity, absenteeism, and early retirement[425].

Based on their inheritance patterns, monogenic disorders can be categorized as sex chromosome-linked, autosomal recessive, or autosomal dominant. Autosomal dominant

monogenic disorders can be further categorized as haploinsufficient (where a LOF mutation leads to decreased protein levels that are insufficient to maintain normal function), dominant-negative (the mutation results in a toxic gain of function phenotype), or they can belong to both categories. While most monogenic disorders manifest due to LOF mutations, the focus of therapeutic development has mainly been on targeted reduction of toxic gain of function proteins or blocking protein interactions[218,285,377]. A substantial subset of human diseases can be addressed by targeted upregulation of protein levels.

Genetic medicine technologies hold incredible potential to resolve these previously intractable disease conditions. The ever-increasing understanding of disease mechanisms and rapid advancements in oligonucleotide (ON) platforms is beginning to translate into success in the clinic[218]. ON technologies are capable of directly engaging molecular targets to modulate gene expression through a diverse set of mechanisms including enzymatic RNA degradation through the RNAi pathway or RNase H recruitment, steric blocking and splicing modulation, DNA and RNA editing, and non-Watson-Crick interactions with proteins and their post-translational derivatives[218,219]. The development of ON platforms provides multiple new avenues for "drugging the undruggable" disease targets, as their mechanisms of action are ideal for targeting monogenic disease conditions[377]. The recent successes of ON therapeutics are a result of almost four decades of research. During this period, advancements in the field of ON chemistry have enabled significant improvements in ON pharmacokinetic and pharmacodynamic properties[218-220].

Several approaches to utilize ONs for targeted gene upregulation are currently being evaluated in the clinic. Stoke Therapeutics (Bedford, MA, USA) have described a splice switching ON platform called TANGO (targeted augmentation of nuclear gene output), that aims to correct unproductive splicing as a strategy to restore the pool of translatable mRNA in the cytoplasm[286,287]. Unproductive mRNA isoforms, which are ultimately degraded via the

nonsense-mediated decay (NMD) pathway, due to the presence of premature stop codon (PTC)-containing "poison exons", are estimated to represent a third of all spliced events[426]. An alternative strategy to the TANGO approach is being evaluated by MiNA Therapeutics (London, UK). Their platform employs small activating RNAs (saRNAs) that target promoter regions to increase transcriptional efficiency of target mRNAs[383-386]. Similarly, but on a post-transcriptional level, CAMP4 Therapeutics (Cambridge, MA, USA) are using ONs to target regulatory RNAs (regRNAs), including enhancer RNA (eRNA), promoter-associated RNA (paRNA), and non-coding RNA (ncRNA), for the purpose of targeted mRNA expression upregulation[427]. While these efforts are currently ongoing, no gene activation therapeutics have been approved to date.

The work presented in this thesis supports the disruption of uORF initiation codons as a viable strategy for targeted gene upregulation. By targeting haploinsufficient and tumor suppressor genes, this approach can be immediately beneficial for therapeutic development in monogenic disorders and cancer. uORF disruption can additionally be utilized to activate disease modifier genes such as the neuroprotective *BDNF* , which may be relevant in the treatment of Alzheimer's disease, Parkinson's disease, multiple sclerosis, and Huntington's disease[349].

In addition to targeted gene activation, uORFs may also be introduced to attenuate or titrate expression of the pORF; a strategy which may be utilized to downregulate aberrant expression of proto-oncogenes. Finally, observations in the *GCN4* and *ATF4* genes suggest that uORF architectures can be used to engineer systems where the pORF is only expressed during stress conditions[169-173]. Importantly, the post-transcriptional targeting of uORFs at an RNA level is orthogonal to the TANGO, saRNA, and regRNA approaches which opens the possibility to further enhance protein upregulation through a combination therapy strategy.

The development of the *uORFeome* database, and the *uORF Detective* tool enables the analysis of uORFs in the context of their structural features and translational activity. A computational analysis of proteomics data across 29 human tissues revealed that uORF position and the number of uORFs in the 5′ UTR are strong predictors of translational repression of the pORF.

uORF position influences the ability of post-translating 40S ribosomes to regain initiation competency before they reach the pORF[1,162,163,194]. It is likely due to this feature, that the uORFs of the neuromuscular disease target genes *SMN2* and *UTRN*, which are located near the 5′ cap structure, were found to not influence pORF expression in their native context. However, when relocated in proximity to the pORF and placed under an optimal Kozak context, the *SMN2* uORF was found to inhibit translation of the pORF. In relation to this observation, ribosome profiling data shows strong initiation at the cap-proximal minimal uORF 0 of the *ATF4* gene. However, this uORF was found to also not repress translation of the pORF. These results may highlight a previously unappreciated ability of the 40S ribosome to stay engaged with the mRNA after encountering a uORF termination codon with great efficiency. The fact that no effect is observed when cap-proximal uORFs initiation codons are disrupted, may suggest that most 40S ribosomes continue to scan after translating uORFs, and are likely to regain initiation competency provided that there is sufficient sequence distance to the pORF. The exact distance may vary depending on the availability of a ternary complex, 5′ UTR secondary structure, or the acquisition of other necessary initiation factors.

A comprehensive mutagenesis study of the complex multi-uORF 5′ UTR architectures of the *BRD3*, *BDNF*, *SCN1A* genes reveals that there is a redundancy in multi-uORF 5′ UTRs, and that the repression of the pORF is not arising from the activity of a single uORF. An additive derepression effect was observed when consecutive uORFs were silenced from the 3′ end, further highlighting uORF position in the 5′ UTR as an important predictive factor of

repressive activity. Most importantly, it was found that even in genes with complex multi-uORF 5′ UTR architectures, silencing of specific single uORFs may still be of therapeutic value.

uORFs may be optimal targeting regions for genome editing therapies. Single guide CRISPR editing approaches have been applied to targeting uORFs in plants and human cells[373-375]. However, due to the permanent aspect of genome editing, editing specificity is a significant safety concern for translational research[419]. Sequence similarity in the genome, may lead to unanticipated off-target editing, which may result in detrimental clinical effects. In contrast to genome editing, ON treatment effects are transient, titratable, and reversible.

ONs capable of recruiting ADAR to catalyze a targeted A-to-I deamination at uORF initiation codons represent a promising platform for targeting uORFs. This work demonstrates a proof-of-concept for endogenous editing of uORF initiation codons. Given that the ON RNA editing field is in its infancy, the search for an optimal editing ON architecture and chemistry is still ongoing. Endogenous RNA editing has been achieved with a diverse set of ON triggers including unmodified long single stranded RNA, double-stranded stem loop RNA, single stranded chemically modified RNA, and circular RNA[297-300]. RNA editing can be used to silence uORF initiation codons by modifying AUG-to-GUG, but also to silence stop codons with UAA-to-UGG, UGA-to-UGG, or UAG-to-UGG modification, or introduce a uORF with an AUA-to-AUG modification. Steric blocking ONs have been used to upregulate pORF translation by masking uORF initiation codons[216]. However, we were unable to reproduce this effect in multiple experiments. The application of steric block ONs for the purpose of upregulation is contradictory to their already established mode of action as translational blockers. Steric blocking ONs have been utilized extensively by the zebrafish community as a tool for gene knockdown, and their ability to block translation has been demonstrated in mice, rats, and pigs[428-432]. Steric block ONs have additionally been tested as translational inhibitors

of c-MYC and the influenza virus in two separate clinical trials[378,433,434]. Yet, a recent study where steric blocking ONs were utilized to target the initial segment of the 5′ UTR of the *SMN2* gene, reported an increase in SMN translation[411]. While these ONs also mask the *SMN2* uORF initiation codon, it was experimentally shown in this thesis that the *SMN2* uORF is inactive and inconsequential to translation of the pORF. Therefore, it was proposed that the observed SMN upregulation effect may be a consequence of mRNA stabilization[411]. Although, it may also be possible that the effect was observed due to disruption of secondary structures near the 5′ cap, which may result in enhanced 40S recruitment. Currently, there is more support in literature for steric block ONs as translational blockers than translational activators. Additional mechanistic studies are required to better understand the potential dual mode of action of steric blocking ONs designed to target regions in the 5′ UTR.

The portfolio of mechanisms by which ONs can engage various molecules in the cell is growing, and now includes genome editing, RNA editing, splice correction, steric blocking, protein and post-translational derivative interaction, and enzymatic degradation. Each of these platforms have been extensively validated *in vitro*, but their broad applicability *in vivo* is limited by the inherent chemical properties of ONs[218]. The key challenge in the development of ON therapeutics remains delivery and biodistribution. Solving the ON delivery problem for additional tissues can unlock the full potential of ON therapeutics for treating genetic diseases.

According to a recent analysis the total amount of generated genomics data doubles approximately every 7 months[435]. The vastness and complexity of this data presents a significant analytical challenge, but also provides an opportunity for novel approaches in drug development. Artificial intelligence can be applied to untangle the complexity of biological data, and has the potential to hasten drug discovery, shed light on the molecular and genomic features that are critical for the design of better precision medicines, and ultimately improve the quality of life of patients afflicted by genetic conditions[436-438].

Artificial intelligence can have profound impact in biotechnology, as machine learning models are capable of recognizing data patterns that may not be obvious in typical laboratory experiments performed at a smaller scale[439]. One such example was demonstrated in this thesis, with the discovery and validation of an inhibitory motif present in the uORF encoded sequence. Conversely, an *in vitro* search of such motifs would be laborious, costly, time-consuming, and due to algorithm space complexity, never be as comprehensive. More generally, drug discovery requires the mining of large sets of biological data, which are too complex to be comprehensible for human analysis. Harnessing the power of AI can help navigate this complexity and generate actionable predictions which can be tested in the laboratory. For example, with enough training data from high-throughput *in vitro* experiments, machine learning models may be trained to predict optimal ON chemistry architectures, provide insights into the best regions to target mRNA, and even simulate the effects of treatment *in silico*. Ribosome profiling data may be used to train models to predict translation in the 5′ UTR for regions with low coverage. Such models can be utilized to estimate uORF activity, but may also be used to predict the behavior and location of ribosomes when uORFs are disrupted.

In conclusion, uORFs are critical post-transcriptional *cis*-regulatory elements of translation, which are amenable to targeting with ON technologies. Due to their prevalence in the transcriptome, targeting uORFs can be a viable strategy for gene activation in numerous genetic disorders. Computational analyses reveal uORF position and uORF count as critical features for predicting uORF effects on the translation of the pORF. Significant upregulation was observed when uORF initiation codons were mutated in genes associated with cancer and neuromuscular disease. Despite redundancy, genes with complex multi-uORF 5′ UTR architectures may still have therapeutically relevant targeting uORFs. ON therapeutics, and in particular novel RNA editing platforms can be used to specifically target uORF initiation

codons, uORF stop codons, or introduce uORFs for the purpose of gene attenuation. These findings, and the notion that there is a significant unmet need for the treatment of monogenic disorders, supports future exploration of uORFs as promising therapeutic targets.

7 References

1. Barbosa, C., I. Peixeiro, and L. Romao, *Gene expression regulation by upstream open reading frames and human disease.* PLoS Genet, 2013. **9**(8): p. e1003529.
2. Laursen, B.S., et al., *Initiation of Protein Synthesis in Bacteria.* Microbiology and Molecular Biology Reviews, 2005. **69**(1): p. 101-123.
3. Kudla, G., et al., *Coding-Sequence Determinants of Gene Expression in Escherichia coli.* Science, 2009. **324**(5924): p. 255-258.
4. Kahvejian, A., et al., *Mammalian poly(A)-binding protein is a eukaryotic translation initiation factor, which acts via multiple mechanisms.* Genes & Development, 2005. **19**(1): p. 104-113.
5. 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.
6. Kapp, L.D. and J.R. Lorsch, *The molecular mechanics of eukaryotic translation.* Annu Rev Biochem, 2004. **73**: p. 657-704.
7. Jackson, R.J., C.U.T. Hellen, and T.V. Pestova, *The mechanism of eukaryotic translation initiation and principles of its regulation.* Nature Reviews Molecular Cell Biology, 2010. **11**(2): p. 113-127.
8. Grifo, J.A., et al., *New initiation factor activity required for globin mRNA translation.* J Biol Chem, 1983. **258**(9): p. 5804-10.
9. Wang, Z. and M. Kiledjian, *The Poly(A)-Binding Protein and an mRNA Stability Protein Jointly Regulate an Endoribonuclease Activity.* Molecular and Cellular Biology, 2000. **20**(17): p. 6334-6341.
10. Kajjo, S., et al., *PABP prevents the untimely decay of select mRNA populations in human cells.* EMBO J, 2022. **41**(6): p. e108650.
11. des Georges, A., et al., *Structure of mammalian eIF3 in the context of the 43S preinitiation complex.* Nature, 2015. **525**(7570): p. 491-5.
12. Zhou, M., et al., *Mass spectrometry reveals modularity and a complete subunit interaction map of the eukaryotic translation factor eIF3.* Proc Natl Acad Sci U S A, 2008. **105**(47): p. 18139-44.
13. Walsh, D. and I. Mohr, *Coupling 40S ribosome recruitment to modification of a cap-binding initiation factor by eIF3 subunit e.* Genes & Development, 2014. **28**(8): p. 835-840.
14. Hinnebusch, A.G., *eIF3: a versatile scaffold for translation initiation complexes.* Trends Biochem Sci, 2006. **31**(10): p. 553-62.
15. Sonenberg, N. and A.G. Hinnebusch, *Regulation of translation initiation in eukaryotes: mechanisms and biological targets.* Cell, 2009. **136**(4): p. 731-45.
16. Hernandez, G., V.G. Osnaya, and X. Perez-Martinez, *Conservation and Variability of the AUG Initiation Codon Context in Eukaryotes.* Trends Biochem Sci, 2019. **44**(12): p. 1009-1021.
17. Shin, B.-S., et al., *Initiation factor eIF2γ promotes eIF2–GTP–Met-tRNAiMet ternary complex binding to the 40S ribosome.* Nature Structural & Molecular Biology, 2011. **18**(11): p. 1227-1234.

18. Yu, Y., et al., *Position of eukaryotic translation initiation factor eIF1A on the 40S ribosomal subunit mapped by directed hydroxyl radical probing.* Nucleic acids research, 2009. **37**(15): p. 5167-5182.
19. Saini, A.K., et al., *Regulatory elements in eIF1A control the fidelity of start codon selection by modulating tRNAiMet binding to the ribosome.* Genes & development, 2010. **24**(1): p. 97-110.
20. Jose, et al., *Conformational Differences between Open and Closed States of the Eukaryotic Translation Initiation Complex.* Molecular Cell, 2015. **59**(3): p. 399-412.
21. Paulin, F.E.M., et al., *Eukaryotic translation initiation factor 5 (eIF5) acts as a classical GTPase-activator protein.* Current Biology, 2001. **11**(1): p. 55-59.
22. Mathews, M., N. Sonenberg, and J.W.B. Hershey, *Translational control in biology and medicine*. 3rd ed. Cold Spring Harbor monograph series. 2007, Cold Spring Harbor, N.Y.: Cold Spring Harbor Laboratory Press. x, 934 p.
23. Algire, M.A., D. Maag, and J.R. Lorsch, *Pi Release from eIF2, Not GTP Hydrolysis, Is the Step Controlled by Start-Site Selection during Eukaryotic Translation Initiation.* Molecular Cell, 2005. **20**(2): p. 251-262.
24. Valasek, L., et al., *Interactions of eukaryotic translation initiation factor 3 (eIF3) subunit NIP1/c with eIF1 and eIF5 promote preinitiation complex assembly and regulate start codon selection.* Mol Cell Biol, 2004. **24**(21): p. 9437-55.
25. Chakrabarti, A. and U. Maitra, *Function of eukaryotic initiation factor 5 in the formation of an 80 S ribosomal polypeptide chain initiation complex.* J Biol Chem, 1991. **266**(21): p. 14039-45.
26. Hinnebusch, A.G. and J.R. Lorsch, *The Mechanism of Eukaryotic Translation Initiation: New Insights and Challenges.* Cold Spring Harbor Perspectives in Biology, 2012. **4**(10): p. a011544-a011544.
27. Voigts-Hoffmann, F., S. Klinge, and N. Ban, *Structural insights into eukaryotic ribosomes and the initiation of translation.* Current opinion in structural biology, 2012. **22**(6): p. 768-777.
28. Hinnebusch, A.G., *The scanning mechanism of eukaryotic translation initiation.* Annual review of biochemistry, 2014. **83**: p. 779-812.
29. Kozak, M. and A.J. Shatkin, *Migration of 40 S ribosomal subunits on messenger RNA in the presence of edeine.* Journal of Biological Chemistry, 1978. **253**(18): p. 6568-6577.
30. Kozak, M., *Circumstances and mechanisms of inhibition of translation by secondary structure in eucaryotic mRNAs.* Mol Cell Biol, 1989. **9**(11): p. 5134-42.
31. Kozak, M., *Possible role of flanking nucleotides in recognition of the AUG initiator codon by eukaryotic ribosomes.* Nucleic Acids Res, 1981. **9**(20): p. 5233-52.
32. Kozak, M., *Compilation and analysis of sequences upstream from the translational start site in eukaryotic mRNAs.* Nucleic Acids Res, 1984. **12**(2): p. 857-72.
33. Kozak, M., *An analysis of 5'-noncoding sequences from 699 vertebrate messenger RNAs.* Nucleic Acids Res, 1987. **15**(20): p. 8125-48.
34. Kozak, M., *Point mutations define a sequence flanking the AUG initiator codon that modulates translation by eukaryotic ribosomes.* Cell, 1986. **44**(2): p. 283-92.
35. Kozak, M., *Pushing the limits of the scanning mechanism for initiation of translation.* Gene, 2002. **299**(1-2): p. 1-34.
36. Kozak, M., *At least six nucleotides preceding the AUG initiator codon enhance translation in mammalian cells.* J Mol Biol, 1987. **196**(4): p. 947-50.

37. Kozak, M., *Point mutations close to the AUG initiator codon affect the efficiency of translation of rat preproinsulin in vivo.* Nature, 1984. **308**(5956): p. 241-246.
38. Grzegorski, S.J., et al., *Natural variability of Kozak sequences correlates with function in a zebrafish model.* PLoS One, 2014. **9**(9): p. e108475.
39. Nakagawa, S., et al., *Diversity of preferred nucleotide sequences around the translation initiation codon in eukaryote genomes.* Nucleic Acids Res, 2008. **36**(3): p. 861-71.
40. Kochetov, A.V., et al., *Eukaryotic mRNAs encoding abundant and scarce proteins are statistically dissimilar in many structural features.* FEBS Lett, 1998. **440**(3): p. 351-5.
41. Hernández, G., V.G. Osnaya, and X. Pérez-Martínez, *Conservation and Variability of the AUG Initiation Codon Context in Eukaryotes.* Trends in Biochemical Sciences, 2019. **44**(12): p. 1009-1021.
42. Kozak, M., *Context effects and inefficient initiation at non-AUG codons in eucaryotic cell-free translation systems.* Mol Cell Biol, 1989. **9**(11): p. 5073-80.
43. Kozak, M., *Structural features in eukaryotic mRNAs that modulate the initiation of translation.* J Biol Chem, 1991. **266**(30): p. 19867-70.
44. Meijer, H.A. and A.A. Thomas, *Control of eukaryotic protein synthesis by upstream open reading frames in the 5'-untranslated region of an mRNA.* Biochem J, 2002. **367**(Pt 1): p. 1-11.
45. Morlé, F., et al., *alpha-Thalassaemia associated with the deletion of two nucleotides at position -2 and -3 preceding the AUG codon.* The EMBO Journal, 1985. **4**(5): p. 1245-1250.
46. Morle, F., J. Starck, and J. Godet, *Alpha-thalassemia due to the deletion of nucleotides -2 and -3 preceding the AUG initiation codon affects translation efficiency both in vitro and in vivo.* Nucleic Acids Res, 1986. **14**(8): p. 3279-92.
47. Jacobson, E.M., et al., *A Graves' Disease-Associated Kozak Sequence Single-Nucleotide Polymorphism Enhances the Efficiency of CD40 Gene Translation: A Case for Translational Pathophysiology.* Endocrinology, 2005. **146**(6): p. 2684-2691.
48. Afshar-Kharghan, V., et al., *Kozak sequence polymorphism of the glycoprotein (GP) Ibalpha gene is a major determinant of the plasma membrane levels of the platelet GP Ib-IX-V complex.* Blood, 1999. **94**(1): p. 186-91.
49. Usuki, F. and K. Maruyama, *Ataxia caused by mutations in the alpha-tocopherol transfer protein gene.* J Neurol Neurosurg Psychiatry, 2000. **69**(2): p. 254-6.
50. Signori, E., et al., *A somatic mutation in the 5'UTR of BRCA1 gene in sporadic breast cancer causes down-modulation of translation efficiency.* Oncogene, 2001. **20**(33): p. 4596-600.
51. Shah, P., et al., *Rate-Limiting Steps in Yeast Protein Translation.* Cell, 2013. **153**(7): p. 1589-1601.
52. Sharma, A.K., et al., *A chemical kinetic basis for measuring translation initiation and elongation rates from ribosome profiling data.* PLOS Computational Biology, 2019. **15**(5): p. e1007070.
53. Teleman, A.A., Y.-W. Chen, and S.M. Cohen, *4E-BP functions as a metabolic brake used under stress conditions but not during normal growth.* Genes & Development, 2005. **19**(16): p. 1844-1848.
54. Walsh, D. and I. Mohr, *Coupling 40S ribosome recruitment to modification of a cap-binding initiation factor by eIF3 subunit e.* Genes Dev, 2014. **28**(8): p. 835-40.
55. Hinnebusch, A.G., et al., *Translational control in biology and medicine.* 2007.

56. Acevedo, J.M., et al., *Changes in global translation elongation or initiation rates shape the proteome via the Kozak sequence.* Scientific Reports, 2018. **8**(1).
57. Noderer, W.L., et al., *Quantitative analysis of mammalian translation initiation sites by FACS-seq.* Mol Syst Biol, 2014. **10**: p. 748.
58. Andreev, D.E., et al., *Non-AUG translation initiation in mammals.* Genome Biology, 2022. **23**(1).
59. Peabody, D.S., *Translation initiation at non-AUG triplets in mammalian cells.* J Biol Chem, 1989. **264**(9): p. 5031-5.
60. Liang, H., et al., *PTENbeta is an alternatively translated isoform of PTEN that regulates rDNA transcription.* Nat Commun, 2017. **8**: p. 14771.
61. Ingolia, N.T., et al., *Genome-Wide Analysis in Vivo of Translation with Nucleotide Resolution Using Ribosome Profiling.* Science, 2009. **324**(5924): p. 218-223.
62. Lee, S., et al., *Global mapping of translation initiation sites in mammalian cells at single-nucleotide resolution.* Proc Natl Acad Sci U S A, 2012. **109**(37): p. E2424-32.
63. Belinky, F., I.B. Rogozin, and E.V. Koonin, *Selection on start codons in prokaryotes and potential compensatory nucleotide substitutions.* Scientific Reports, 2017. **7**(1).
64. Schneider-Poetsch, T., et al., *Inhibition of eukaryotic translation elongation by cycloheximide and lactimidomycin.* Nature Chemical Biology, 2010. **6**(3): p. 209-217.
65. Poehlsgaard, J. and S. Douthwaite, *The bacterial ribosome as a target for antibiotics.* Nat Rev Microbiol, 2005. **3**(11): p. 870-81.
66. Ingolia, N.T., et al., *The ribosome profiling strategy for monitoring translation in vivo by deep sequencing of ribosome-protected mRNA fragments.* Nature Protocols, 2012. **7**(8): p. 1534-1550.
67. Ju, J., et al., *Lactimidomycin, iso-migrastatin and related glutarimide-containing 12-membered macrolides are extremely potent inhibitors of cell migration.* J Am Chem Soc, 2009. **131**(4): p. 1370-1.
68. Sugawara, K., et al., *Lactimidomycin, a new glutarimide group antibiotic. Production, isolation, structure and biological activity.* J Antibiot (Tokyo), 1992. **45**(9): p. 1433-41.
69. Fritsch, C., et al., *Genome-wide search for novel human uORFs and N-terminal protein extensions using ribosomal footprinting.* Genome Res, 2012. **22**(11): p. 2208-18.
70. Nicholas, Liana, and Jonathan, *Ribosome Profiling of Mouse Embryonic Stem Cells Reveals the Complexity and Dynamics of Mammalian Proteomes.* Cell, 2011. **147**(4): p. 789-802.
71. Kearse, M.G. and J.E. Wilusz, *Non-AUG translation: a new start for protein synthesis in eukaryotes.* Genes & Development, 2017. **31**(17): p. 1717-1731.
72. Hann, S.R., et al., *A non-AUG translational initiation in c-myc exon 1 generates an N-terminally distinct protein whose synthesis is disrupted in Burkitt's lymphomas.* Cell, 1988. **52**(2): p. 185-95.
73. Loughran, G., et al., *Unusually efficient CUG initiation of an overlapping reading frame in POLG mRNA yields novel protein POLGARF.* Proceedings of the National Academy of Sciences, 2020. **117**(40): p. 24936-24946.
74. Alexander, J., W.L. Noderer, and C.L. Wang, *Complete motif analysis of sequence requirements for translation initiation at non-AUG start codons.* Nucleic Acids Research, 2018. **46**(2): p. 985-994.
75. Hutten, S. and D. Dormann, *RAN translation down.* Nature Neuroscience, 2019. **22**(9): p. 1379-1380.

76. Peter, et al., *Unconventional Translation of C9ORF72 GGGGCC Expansion Generates Insoluble Polypeptides Specific to c9FTD/ALS.* Neuron, 2013. **77**(4): p. 639-646.
77. Banez-Coronel, M., et al., *RAN Translation in Huntington Disease.* Neuron, 2015. **88**(4): p. 667-77.
78. Scoles, D.R., et al., *Repeat Associated Non-AUG Translation (RAN Translation) Dependent on Sequence Downstream of the ATXN2 CAG Repeat.* PLoS One, 2015. **10**(6): p. e0128769.
79. Cleary, J.D. and L.P.W. Ranum, *Repeat-associated non-ATG (RAN) translation in neurological disease.* Human Molecular Genetics, 2013. **22**(R1): p. R45-R51.
80. Zu, T., et al., *RAN Translation Regulated by Muscleblind Proteins in Myotonic Dystrophy Type 2.* Neuron, 2017. **95**(6): p. 1292-1305 e5.
81. Green, K.M., et al., *RAN translation at C9orf72-associated repeat expansions is selectively enhanced by the integrated stress response.* Nat Commun, 2017. **8**(1): p. 2005.
82. Kearse, M.G. and J.E. Wilusz, *Non-AUG translation: a new start for protein synthesis in eukaryotes.* Genes Dev, 2017. **31**(17): p. 1717-1731.
83. Sanz, M.A., et al., *A viral RNA motif involved in signaling the initiation of translation on non-AUG codons.* RNA, 2019. **25**(4): p. 431-452.
84. Pisareva, V.P., A.V. Pisarev, and I.S. Fernandez, *Dual tRNA mimicry in the Cricket Paralysis Virus IRES uncovers an unexpected similarity with the Hepatitis C Virus IRES.* Elife, 2018. **7**.
85. Fernandez, I.S., et al., *Initiation of translation by cricket paralysis virus IRES requires its translocation in the ribosome.* Cell, 2014. **157**(4): p. 823-31.
86. Jan, E. and P. Sarnow, *Factorless ribosome assembly on the internal ribosome entry site of cricket paralysis virus.* J Mol Biol, 2002. **324**(5): p. 889-902.
87. Kozak, M., *Initiation of translation in prokaryotes and eukaryotes.* Gene, 1999. **234**(2): p. 187-208.
88. Yang, Y. and Z. Wang, *IRES-mediated cap-independent translation, a path leading to hidden proteome.* Journal of Molecular Cell Biology, 2019. **11**(10): p. 911-919.
89. Mailliot, J. and F. Martin, *Viral internal ribosomal entry sites: four classes for one goal.* Wiley Interdiscip Rev RNA, 2018. **9**(2).
90. Lloyd, R.E., *Translational control by viral proteinases.* Virus Res, 2006. **119**(1): p. 76-88.
91. Komarova, A.V., A.L. Haenni, and B.C. Ramirez, *Virus versus host cell translation love and hate stories.* Adv Virus Res, 2009. **73**: p. 99-170.
92. Jang, S.K., et al., *A segment of the 5' nontranslated region of encephalomyocarditis virus RNA directs internal entry of ribosomes during in vitro translation.* J Virol, 1988. **62**(8): p. 2636-43.
93. Pelletier, J. and N. Sonenberg, *Internal initiation of translation of eukaryotic mRNA directed by a sequence derived from poliovirus RNA.* Nature, 1988. **334**(6180): p. 320-5.
94. Komar, A.A. and M. Hatzoglou, *Cellular IRES-mediated translation: the war of ITAFs in pathophysiological states.* Cell Cycle, 2011. **10**(2): p. 229-40.
95. Yang, Y., et al., *Extensive translation of circular RNAs driven by N(6)-methyladenosine.* Cell Res, 2017. **27**(5): p. 626-641.
96. Yang, Y. and Z. Wang, *IRES-mediated cap-independent translation, a path leading to hidden proteome.* J Mol Cell Biol, 2019. **11**(10): p. 911-919.

97. Weingarten-Gabbay, S., et al., *Comparative genetics. Systematic discovery of cap-independent translation sequences in human and viral genomes.* Science, 2016. **351**(6270).
98. Chappell, S.A., et al., *Ribosomal shunting mediated by a translational enhancer element that base pairs to 18S rRNA.* Proc Natl Acad Sci U S A, 2006. **103**(25): p. 9488-93.
99. Mauro, V.P., S.A. Chappell, and J. Dresios, *Analysis of ribosomal shunting during translation initiation in eukaryotic mRNAs.* Methods Enzymol, 2007. **429**: p. 323-54.
100. Futterer, J., et al., *Positive and negative control of translation by the leader sequence of cauliflower mosaic virus pregenomic 35S RNA.* EMBO J, 1990. **9**(6): p. 1697-707.
101. Futterer, J., Z. Kiss-Laszlo, and T. Hohn, *Nonlinear ribosome migration on cauliflower mosaic virus 35S RNA.* Cell, 1993. **73**(4): p. 789-802.
102. Kwan, T. and S.R. Thompson, *Noncanonical Translation Initiation in Eukaryotes.* Cold Spring Harbor Perspectives in Biology, 2019. **11**(4): p. a032672.
103. Yang, T.H., et al., *Human IRES Atlas: an integrative platform for studying IRES-driven translational regulation in humans.* Database (Oxford), 2021. **2021**.
104. Sharathchandra, A., A. Katoch, and S. Das, *IRES mediated translational regulation of p53 isoforms.* Wiley Interdiscip Rev RNA, 2014. **5**(1): p. 131-9.
105. Dever, T.E., J.D. Dinman, and R. Green, *Translation Elongation and Recoding in Eukaryotes.* Cold Spring Harbor Perspectives in Biology, 2018. **10**(8): p. a032649.
106. Rodnina, M.V. and W. Wintermeyer, *Ribosome fidelity: tRNA discrimination, proofreading and induced fit.* Trends Biochem Sci, 2001. **26**(2): p. 124-30.
107. Munro, J.B., et al., *Identification of two distinct hybrid state intermediates on the ribosome.* Mol Cell, 2007. **25**(4): p. 505-17.
108. Matassova, A.B., M.V. Rodnina, and W. Wintermeyer, *Elongation factor G-induced structural change in helix 34 of 16S rRNA related to translocation on the ribosome.* RNA, 2001. **7**(12): p. 1879-85.
109. Rodnina, M.V., et al., *[Mechanism of tRNA translocation on the ribosome].* Mol Biol (Mosk), 2001. **35**(4): p. 655-65.
110. Joana, J. Racle, and V. Hatzimanikatis, *Analysis of Translation Elongation Dynamics in the Context of an Escherichia coli Cell.* Biophysical Journal, 2016. **110**(9): p. 2120-2131.
111. Zouridis, H. and V. Hatzimanikatis, *Effects of codon distributions and tRNA competition on protein translation.* Biophys J, 2008. **95**(3): p. 1018-33.
112. Fluitt, A., E. Pienaar, and H. Viljoen, *Ribosome kinetics and aa-tRNA competition determine rate and fidelity of peptide synthesis.* Comput Biol Chem, 2007. **31**(5-6): p. 335-46.
113. Riba, A., et al., *Protein synthesis rates and ribosome occupancies reveal determinants of translation elongation rates.* Proc Natl Acad Sci U S A, 2019. **116**(30): p. 15023-15032.
114. Schuller, A.P., et al., *eIF5A Functions Globally in Translation Elongation and Termination.* Molecular Cell, 2017. **66**(2): p. 194-205.e5.
115. Gerashchenko, M.V., et al., *Translation elongation rate varies among organs and decreases with age.* Nucleic Acids Research, 2021. **49**(2): p. e9-e9.
116. Lind, C., A. Oliveira, and J. Åqvist, *Origin of the omnipotence of eukaryotic release factor 1.* Nature Communications, 2017. **8**(1).

117. Song, H., et al., *The Crystal Structure of Human Eukaryotic Release Factor eRF1—Mechanism of Stop Codon Recognition and Peptidyl-tRNA Hydrolysis.* Cell, 2000. **100**(3): p. 311-321.
118. Pisarev, A.V., et al., *The role of ABCE1 in eukaryotic posttermination ribosomal recycling.* Mol Cell, 2010. **37**(2): p. 196-210.
119. Young, D.J., et al., *Rli1/ABCE1 Recycles Terminating Ribosomes and Controls Translation Reinitiation in 3'UTRs In Vivo.* Cell, 2015. **162**(4): p. 872-84.
120. Young, D.J. and N.R. Guydosh, *Hcr1/eIF3j Is a 60S Ribosomal Subunit Recycling Accessory Factor In Vivo.* Cell Rep, 2019. **28**(1): p. 39-50 e4.
121. Hellen, C.U.T., *Translation Termination and Ribosome Recycling in Eukaryotes.* Cold Spring Harbor Perspectives in Biology, 2018. **10**(10): p. a032656.
122. Lomakin, I.B., et al., *The fidelity of translation initiation: reciprocal activities of eIF1, IF3 and YciH.* EMBO J, 2006. **25**(1): p. 196-210.
123. Skabkin, M.A., et al., *Activities of Ligatin and MCT-1/DENR in eukaryotic translation initiation and ribosomal recycling.* Genes Dev, 2010. **24**(16): p. 1787-801.
124. Ingolia, N.T., *Ribosome Footprint Profiling of Translation throughout the Genome.* Cell, 2016. **165**(1): p. 22-33.
125. Crick, F., *Central Dogma of Molecular Biology.* Nature, 1970. **227**(5258): p. 561-563.
126. Liu, Y., A. Beyer, and R. Aebersold, *On the Dependency of Cellular Protein Levels on mRNA Abundance.* Cell, 2016. **165**(3): p. 535-50.
127. Ghaemmaghami, S., et al., *Global analysis of protein expression in yeast.* Nature, 2003. **425**(6959): p. 737-741.
128. Maier, T., M. Güell, and L. Serrano, *Correlation of mRNA and protein in complex biological samples.* FEBS Letters, 2009. **583**(24): p. 3966-3973.
129. Mortazavi, A., et al., *Mapping and quantifying mammalian transcriptomes by RNA-Seq.* Nature Methods, 2008. **5**(7): p. 621-628.
130. Zhao, S., et al., *Comparison of RNA-Seq and Microarray in Transcriptome Profiling of Activated T Cells.* PLoS ONE, 2014. **9**(1): p. e78644.
131. Jiang, L., et al., *A Quantitative Proteome Map of the Human Body.* Cell, 2020. **183**(1): p. 269-283.e19.
132. Wang, D., et al., *A deep proteome and transcriptome abundance atlas of 29 healthy human tissues.* Molecular Systems Biology, 2019. **15**(2): p. e8503.
133. Washburn, M.P., D. Wolters, and J.R. Yates, *Large-scale analysis of the yeast proteome by multidimensional protein identification technology.* Nature Biotechnology, 2001. **19**(3): p. 242-247.
134. Zhang, J., et al., *RADAR: annotation and prioritization of variants in the post-transcriptional regulome of RNA-binding proteins.* Genome Biol, 2020. **21**(1): p. 151.
135. Velazquez-Cruz, A., et al., *Post-translational Control of RNA-Binding Proteins and Disease-Related Dysregulation.* Front Mol Biosci, 2021. **8**: p. 658852.
136. Witten, J.T. and J. Ule, *Understanding splicing regulation through RNA splicing maps.* Trends Genet, 2011. **27**(3): p. 89-97.
137. Bartel, D.P., *MicroRNAs.* Cell, 2004. **116**(2): p. 281-297.
138. Wightman, B., I. Ha, and G. Ruvkun, *Posttranscriptional regulation of the heterochronic gene lin-14 by lin-4 mediates temporal pattern formation in C. elegans.* Cell, 1993. **75**(5): p. 855-62.

139. Lee, R.C., R.L. Feinbaum, and V. Ambros, *The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14.* Cell, 1993. **75**(5): p. 843-54.
140. Ambros, V. and H.R. Horvitz, *The lin-14 locus of Caenorhabditis elegans controls the time of expression of specific postembryonic developmental events.* Genes Dev, 1987. **1**(4): p. 398-414.
141. MacRae, I.J., et al., *In vitro reconstitution of the human RISC-loading complex.* Proc Natl Acad Sci U S A, 2008. **105**(2): p. 512-7.
142. Maniataki, E. and Z. Mourelatos, *A human, ATP-independent, RISC assembly machine fueled by pre-miRNA.* Genes Dev, 2005. **19**(24): p. 2979-90.
143. Gregory, R.I., et al., *Human RISC couples microRNA biogenesis and posttranscriptional gene silencing.* Cell, 2005. **123**(4): p. 631-40.
144. De Fougerolles, A., et al., *Interfering with disease: a progress report on siRNA-based therapeutics.* Nature Reviews Drug Discovery, 2007. **6**(6): p. 443-453.
145. Valinezhad Orang, A., R. Safaralizadeh, and M. Kazemzadeh-Bavili, *Mechanisms of miRNA-Mediated Gene Regulation from Common Downregulation to mRNA-Specific Upregulation.* International Journal of Genomics, 2014. **2014**: p. 1-15.
146. Payne, S.H., *The utility of protein and mRNA correlation.* Trends Biochem Sci, 2015. **40**(1): p. 1-3.
147. Vogel, C. and E.M. Marcotte, *Insights into the regulation of protein abundance from proteomic and transcriptomic analyses.* Nat Rev Genet, 2012. **13**(4): p. 227-32.
148. Li, J.J., G.L. Chew, and M.D. Biggin, *Quantitative principles of cis-translational control by general mRNA sequence features in eukaryotes.* Genome Biol, 2019. **20**(1): p. 162.
149. Pelletier, J. and N. Sonenberg, *Insertion mutagenesis to increase secondary structure within the 5' noncoding region of a eukaryotic mRNA reduces translational efficiency.* Cell, 1985. **40**(3): p. 515-26.
150. Kozak, M., *Influences of mRNA secondary structure on initiation by eukaryotic ribosomes.* Proc Natl Acad Sci U S A, 1986. **83**(9): p. 2850-4.
151. Namy, O., et al., *Reprogrammed genetic decoding in cellular gene expression.* Mol Cell, 2004. **13**(2): p. 157-68.
152. Marzluff, W.F. and K.P. Koreski, *Birth and Death of Histone mRNAs.* Trends in Genetics, 2017. **33**(10): p. 745-759.
153. Zhao, J., et al., *IRESbase: A Comprehensive Database of Experimentally Validated Internal Ribosome Entry Sites.* Genomics Proteomics Bioinformatics, 2020. **18**(2): p. 129-139.
154. Zhou, Z.D. and E.K. Tan, *Iron regulatory protein (IRP)-iron responsive element (IRE) signaling pathway in human neurodegenerative diseases.* Mol Neurodegener, 2017. **12**(1): p. 75.
155. Winzen, R., et al., *Distinct domains of AU-rich elements exert different functions in mRNA destabilization and stabilization by p38 mitogen-activated protein kinase or HuR.* Mol Cell Biol, 2004. **24**(11): p. 4835-47.
156. Fernandes, L.D., A.P.S. Moura, and L. Ciandrini, *Gene length as a regulator for ribosome recruitment and protein synthesis: theoretical insights.* Sci Rep, 2017. **7**(1): p. 17409.
157. Yu, C.H., et al., *Codon Usage Influences the Local Rate of Translation Elongation to Regulate Co-translational Protein Folding.* Mol Cell, 2015. **59**(5): p. 744-54.

158. Hershberg, R. and D.A. Petrov, *Selection on codon bias.* Annu Rev Genet, 2008. **42**: p. 287-99.
159. Xia, X., *Detailed Dissection and Critical Evaluation of the Pfizer/BioNTech and Moderna mRNA Vaccines.* Vaccines (Basel), 2021. **9**(7).
160. Wang, X.Q., *5'-Untranslated regions with multiple upstream AUG codons can support low-level translation via leaky scanning and reinitiation.* Nucleic Acids Research, 2004. **32**(4): p. 1382-1391.
161. Morris, D.R. and A.P. Geballe, *Upstream open reading frames as regulators of mRNA translation.* Mol Cell Biol, 2000. **20**(23): p. 8635-42.
162. Calvo, S.E., D.J. Pagliarini, and V.K. Mootha, *Upstream open reading frames cause widespread reduction of protein expression and are polymorphic among humans.* Proc Natl Acad Sci U S A, 2009. **106**(18): p. 7507-12.
163. Kozak, M., *Effects of intercistronic length on the efficiency of reinitiation by eucaryotic ribosomes.* Mol Cell Biol, 1987. **7**(10): p. 3438-45.
164. Kozak, M., *Selection of initiation sites by eucaryotic ribosomes: effect of inserting AUG triplets upstream from the coding sequence for preproinsulin.* Nucleic Acids Res, 1984. **12**(9): p. 3873-93.
165. Liu, C.C., C.C. Simonsen, and A.D. Levinson, *Initiation of translation at internal AUG codons in mammalian cells.* Nature, 1984. **309**(5963): p. 82-5.
166. Johnstone, T.G., A.A. Bazzini, and A.J. Giraldez, *Upstream ORFs are prevalent translational repressors in vertebrates.* The EMBO Journal, 2016. **35**(7): p. 706-723.
167. McGillivray, P., et al., *A comprehensive catalog of predicted functional upstream open reading frames in humans.* Nucleic Acids Res, 2018. **46**(7): p. 3326-3338.
168. Matsui, M., et al., *Bioinformatic analysis of post-transcriptional regulation by uORF in human and mouse.* FEBS Lett, 2007. **581**(22): p. 4184-8.
169. Lu, P.D., H.P. Harding, and D. Ron, *Translation reinitiation at alternative open reading frames regulates gene expression in an integrated stress response.* J Cell Biol, 2004. **167**(1): p. 27-33.
170. Vasudevan, D., et al., *Translational induction of ATF4 during integrated stress response requires noncanonical initiation factors eIF2D and DENR.* Nature Communications, 2020. **11**(1).
171. Vattem, K.M. and R.C. Wek, *Reinitiation involving upstream ORFs regulates ATF4 mRNA translation in mammalian cells.* Proc Natl Acad Sci U S A, 2004. **101**(31): p. 11269-74.
172. Hinnebusch, A.G., *Translational regulation of GCN4 and the general amino acid control of yeast.* Annu Rev Microbiol, 2005. **59**: p. 407-50.
173. Mueller, P.P. and A.G. Hinnebusch, *Multiple upstream AUG codons mediate translational control of GCN4.* Cell, 1986. **45**(2): p. 201-7.
174. Grant, C.M. and A.G. Hinnebusch, *Effect of sequence context at stop codons on efficiency of reinitiation in GCN4 translational control.* Mol Cell Biol, 1994. **14**(1): p. 606-18.
175. Rogers, G.W., Jr., G.M. Edelman, and V.P. Mauro, *Differential utilization of upstream AUGs in the beta-secretase mRNA suggests that a shunting mechanism regulates translation.* Proc Natl Acad Sci U S A, 2004. **101**(9): p. 2794-9.
176. Lammich, S., et al., *Expression of the Alzheimer protease BACE1 is suppressed via its 5'-untranslated region.* EMBO Rep, 2004. **5**(6): p. 620-5.

177. Lee, D.S.M., et al., *Disrupting upstream translation in mRNAs is associated with human disease.* Nature Communications, 2021. **12**(1).
178. Ingolia, N.T., J.A. Hussmann, and J.S. Weissman, *Ribosome Profiling: Global Views of Translation.* Cold Spring Harb Perspect Biol, 2019. **11**(5).
179. Spealman, P., et al., *Conserved non-AUG uORFs revealed by a novel regression analysis of ribosome profiling data.* Genome Res, 2018. **28**(2): p. 214-222.
180. Neafsey, D.E. and J.E. Galagan, *Dual modes of natural selection on upstream open reading frames.* Mol Biol Evol, 2007. **24**(8): p. 1744-51.
181. Iacono, M., F. Mignone, and G. Pesole, *uAUG and uORFs in human and rodent 5'untranslated mRNAs.* Gene, 2005. **349**: p. 97-105.
182. Chen, H.H. and W.Y. Tarn, *uORF-mediated translational control: recently elucidated mechanisms and implications in cancer.* RNA Biol, 2019. **16**(10): p. 1327-1338.
183. Nyikó, T., et al., *Plant upstream ORFs can trigger nonsense-mediated mRNA decay in a size-dependent manner.* Plant Molecular Biology, 2009. **71**(4-5): p. 367-378.
184. Mendell, J.T., et al., *Nonsense surveillance regulates expression of diverse classes of mammalian transcripts and mutes genomic noise.* Nat Genet, 2004. **36**(10): p. 1073-8.
185. Stockklausner, C., *The uORF-containing thrombopoietin mRNA escapes nonsense-mediated decay (NMD).* Nucleic Acids Research, 2006. **34**(8): p. 2355-2363.
186. Linde, L., et al., *The efficiency of nonsense-mediated mRNA decay is an inherent character and varies among different cells.* Eur J Hum Genet, 2007. **15**(11): p. 1156-62.
187. Dyle, M.C., et al., *How to get away with nonsense: Mechanisms and consequences of escape from nonsense-mediated RNA decay.* Wiley Interdiscip Rev RNA, 2020. **11**(1): p. e1560.
188. Sato, H. and R.H. Singer, *Cellular variability of nonsense-mediated mRNA decay.* Nature Communications, 2021. **12**(1).
189. Kozak, M., *The scanning model for translation: an update.* J Cell Biol, 1989. **108**(2): p. 229-41.
190. Chen, J., et al., *Pervasive functional translation of noncanonical human open reading frames.* Science, 2020. **367**(6482): p. 1140-1146.
191. Poyry, T.A., A. Kaminski, and R.J. Jackson, *What determines whether mammalian ribosomes resume scanning after translation of a short upstream open reading frame?* Genes Dev, 2004. **18**(1): p. 62-75.
192. Mohammad, M.P., et al., *eIF4G is retained on ribosomes elongating and terminating on short upstream ORFs to control reinitiation in yeast.* Nucleic Acids Research, 2021. **49**(15): p. 8743-8756.
193. Luukkonen, B.G., W. Tan, and S. Schwartz, *Efficiency of reinitiation of translation on human immunodeficiency virus type 1 mRNAs is determined by the length of the upstream open reading frame and by intercistronic distance.* J Virol, 1995. **69**(7): p. 4086-94.
194. Kozak, M., *Constraints on reinitiation of translation in mammals.* Nucleic Acids Res, 2001. **29**(24): p. 5226-32.
195. Skabkin, M.A., et al., *Reinitiation and other unconventional posttermination events during eukaryotic translation.* Mol Cell, 2013. **51**(2): p. 249-64.
196. Schleich, S., et al., *DENR-MCT-1 promotes translation re-initiation downstream of uORFs to control tissue growth.* Nature, 2014. **512**(7513): p. 208-212.

197. Schleich, S., et al., *Identification of transcripts with short stuORFs as targets for DENR*MCTS1-dependent translation in human cells.* Sci Rep, 2017. **7**(1): p. 3722.
198. Janich, P., et al., *Ribosome profiling reveals the rhythmic liver translatome and circadian clock regulation by upstream open reading frames.* Genome Res, 2015. **25**(12): p. 1848-59.
199. Zinoviev, A., C.U.T. Hellen, and T.V. Pestova, *Multiple mechanisms of reinitiation on bicistronic calicivirus mRNAs.* Mol Cell, 2015. **57**(6): p. 1059-1073.
200. Mills, E.W., et al., *Dynamic Regulation of a Ribosome Rescue Pathway in Erythroid Cells and Platelets.* Cell Rep, 2016. **17**(1): p. 1-10.
201. Nürenberg-Goloub, E., et al., *Ribosome recycling is coordinated by processive events in two asymmetric ATP sites of ABCE1.* Life Science Alliance, 2018. **1**(3): p. e201800095.
202. Carvunis, A.R., et al., *Proto-genes and de novo gene birth.* Nature, 2012. **487**(7407): p. 370-4.
203. Pérez-Ortín, J.E., V. Tordera, and S. Chávez, *Homeostasis in the Central Dogma of molecular biology: the importance of mRNA instability.* RNA Biology, 2019. **16**(12): p. 1659-1666.
204. Hinnebusch, A.G., *Translational Regulation of Yeast GCN4.* Journal of Biological Chemistry, 1997. **272**(35): p. 21661-21664.
205. You, T., G.M. Coghill, and A.J.P. Brown, *Translational Control of GCN4*. 2013, Springer New York. p. 2293-2296.
206. Amrani, N., et al., *A faux 3'-UTR promotes aberrant termination and triggers nonsense-mediated mRNA decay.* Nature, 2004. **432**(7013): p. 112-8.
207. Rivas, M.A., et al., *Human genomics. Effect of predicted protein-truncating genetic variants on the human transcriptome.* Science, 2015. **348**(6235): p. 666-9.
208. Schlautmann, L.P. and N.H. Gehring, *A Day in the Life of the Exon Junction Complex.* Biomolecules, 2020. **10**(6): p. 866.
209. Kim, K.M., H. Cho, and Y.K. Kim, *The upstream open reading frame of cyclin-dependent kinase inhibitor 1A mRNA negatively regulates translation of the downstream main open reading frame.* Biochem Biophys Res Commun, 2012. **424**(3): p. 469-75.
210. Kumar, R.D., et al., *Selection for or against escape from nonsense mediated decay is a novel signature for the detection of cancer genes.* Cancer Genet, 2021. **258-259**: p. 80-84.
211. Vallverdu-Prats, M., R. Brugada, and M. Alcalde, *Premature Termination Codon in 5' Region of Desmoplakin and Plakoglobin Genes May Escape Nonsense-Mediated Decay through the Reinitiation of Translation.* Int J Mol Sci, 2022. **23**(2).
212. Young, S.K. and R.C. Wek, *Upstream Open Reading Frames Differentially Regulate Gene-specific Translation in the Integrated Stress Response.* Journal of Biological Chemistry, 2016. **291**(33): p. 16927-16935.
213. Sample, P.J., et al., *Human 5' UTR design and variant effect prediction from a massively parallel translation assay.* Nat Biotechnol, 2019. **37**(7): p. 803-809.
214. Whiffin, N., et al., *The effect of LRRK2 loss-of-function variants in humans.* Nat Med, 2020. **26**(6): p. 869-877.
215. McClements, M.E., et al., *An analysis of the Kozak consensus in retinal genes and its relevance to gene therapy.* Mol Vis, 2021. **27**: p. 233-242.
216. Liang, X.H., et al., *Translation efficiency of mRNAs is increased by antisense oligonucleotides targeting upstream open reading frames.* Nat Biotechnol, 2016. **34**(8): p. 875-80.

217. Liang, X.H., W. Shen, and S.T. Crooke, *Specific Increase of Protein Levels by Enhancing Translation Using Antisense Oligonucleotides Targeting Upstream Open Frames.* Adv Exp Med Biol, 2017. **983**: p. 129-146.
218. Roberts, T.C., R. Langer, and M.J.A. Wood, *Advances in oligonucleotide drug delivery.* Nature Reviews Drug Discovery, 2020. **19**(10): p. 673-694.
219. Smith, C.I.E. and R. Zain, *Therapeutic Oligonucleotides: State of the Art.* Annu Rev Pharmacol Toxicol, 2019. **59**: p. 605-630.
220. Rinaldi, C. and M.J.A. Wood, *Antisense oligonucleotides: the next frontier for treatment of neurological disorders.* Nature Reviews Neurology, 2018. **14**(1): p. 9-21.
221. Eckstein, F., *Phosphorothioates, essential components of therapeutic oligonucleotides.* Nucleic Acid Ther, 2014. **24**(6): p. 374-87.
222. Geary, R.S., et al., *Pharmacokinetics, biodistribution and cell uptake of antisense oligonucleotides.* Adv Drug Deliv Rev, 2015. **87**: p. 46-51.
223. Khvorova, A. and J.K. Watts, *The chemical evolution of oligonucleotide therapies of clinical utility.* Nature Biotechnology, 2017. **35**(3): p. 238-248.
224. Iwamoto, N., et al., *Control of phosphorothioate stereochemistry substantially increases the efficacy of antisense oligonucleotides.* Nat Biotechnol, 2017. **35**(9): p. 845-851.
225. Kandasamy, P., et al., *Impact of guanidine-containing backbone linkages on stereopure antisense oligonucleotides in the CNS.* Nucleic Acids Res, 2022. **50**(10): p. 5401-5423.
226. ClinicalTrials.gov, *Wave Life Sciences clinical trials 2022.* 2022.
227. D'Esposito, R.J., et al., *Challenges with Simulating Modified RNA: Insights into Role and Reciprocity of Experimental and Computational Approaches.* Genes (Basel), 2022. **13**(3).
228. Breiling, A. and F. Lyko, *Epigenetic regulatory functions of DNA modifications: 5-methylcytosine and beyond.* Epigenetics & Chromatin, 2015. **8**(1).
229. Luo, C., P. Hajkova, and J.R. Ecker, *Dynamic DNA methylation: In the right place at the right time.* Science, 2018. **361**(6409): p. 1336-1340.
230. Li, Y. and C. O'Neill, *Persistence of Cytosine Methylation of DNA following Fertilisation in the Mouse.* PLoS ONE, 2012. **7**(1): p. e30687.
231. Mallik, S. and Z. Zhao, *Detecting methylation signatures in neurodegenerative disease by density-based clustering of applications with reducing noise.* Scientific Reports, 2020. **10**(1).
232. Dhuri, K., et al., *Antisense Oligonucleotides: An Emerging Area in Drug Discovery and Development.* Journal of Clinical Medicine, 2020. **9**(6): p. 2004.
233. Hemmi, H., et al., *A Toll-like receptor recognizes bacterial DNA.* Nature, 2000. **408**(6813): p. 740-5.
234. Li, Y. and R.R. Breaker, *Kinetics of RNA degradation by specific base catalysis of transesterification involving the 2 '-hydroxyl group.* Journal of the American Chemical society, 1999. **121**(23): p. 5364-5372.
235. Shen, X. and D.R. Corey, *Chemistry, mechanism and clinical status of antisense oligonucleotides and duplex RNAs.* Nucleic Acids Res, 2018. **46**(4): p. 1584-1600.
236. Monia, B.P., et al., *Evaluation of 2'-modified oligonucleotides containing 2'-deoxy gaps as antisense inhibitors of gene expression.* J Biol Chem, 1993. **268**(19): p. 14514-22.
237. Wong, E. and T. Goldberg, *Mipomersen (kynamro): a novel antisense oligonucleotide inhibitor for the management of homozygous familial hypercholesterolemia.* P T, 2014. **39**(2): p. 119-22.

238. Hair, P., F. Cameron, and K. McKeage, *Mipomersen sodium: first global approval.* Drugs, 2013. **73**(5): p. 487-93.
239. Kim, J., et al., *Patient-Customized Oligonucleotide Therapy for a Rare Genetic Disease.* N Engl J Med, 2019. **381**(17): p. 1644-1652.
240. Biliouris, K., et al., *A Semi-Mechanistic Population Pharmacokinetic Model of Nusinersen: An Antisense Oligonucleotide for the Treatment of Spinal Muscular Atrophy.* CPT: Pharmacometrics & Systems Pharmacology, 2018. **7**(9): p. 581-592.
241. Corey, D.R., *Nusinersen, an antisense oligonucleotide drug for spinal muscular atrophy.* Nature Neuroscience, 2017. **20**(4): p. 497-499.
242. McTigue, P.M., R.J. Peterson, and J.D. Kahn, *Sequence-dependent thermodynamic parameters for locked nucleic acid (LNA)-DNA duplex formation.* Biochemistry, 2004. **43**(18): p. 5388-405.
243. Koshkin, A.A., et al., *LNA (Locked Nucleic Acids): Synthesis of the adenine, cytosine, guanine, 5-methylcytosine, thymine and uracil bicyclonucleoside monomers, oligomerisation, and unprecedented nucleic acid recognition.* Tetrahedron, 1998. **54**(14): p. 3607-3630.
244. Gait, M.J., et al., *Cell-Penetrating Peptide Conjugates of Steric Blocking Oligonucleotides as Therapeutics for Neuromuscular Diseases from a Historical Perspective to Current Prospects of Treatment.* Nucleic Acid Ther, 2019. **29**(1): p. 1-12.
245. Sheng, L., et al., *Comparison of the efficacy of MOE and PMO modifications of systemic antisense oligonucleotides in a severe SMA mouse model.* Nucleic Acids Research, 2020. **48**(6): p. 2853-2865.
246. Lim, K.R., R. Maruyama, and T. Yokota, *Eteplirsen in the treatment of Duchenne muscular dystrophy.* Drug Design, Development and Therapy, 2017. **Volume11**: p. 533-545.
247. Springer, A.D. and S.F. Dowdy, *GalNAc-siRNA Conjugates: Leading the Way for Delivery of RNAi Therapeutics.* Nucleic Acid Ther, 2018. **28**(3): p. 109-118.
248. Debacker, A.J., et al., *Delivery of Oligonucleotides to the Liver with GalNAc: From Research to Registered Therapeutic Drug.* Mol Ther, 2020. **28**(8): p. 1759-1771.
249. Shah, V.N. and L. Pyle, *Lumasiran, an RNAi Therapeutic for Primary Hyperoxaluria Type 1.* N Engl J Med, 2021. **385**(20): p. e69.
250. Samuel, E., et al., *Inclisiran: A First-in-Class siRNA Therapy for Lowering Low-Density Lipoprotein Cholesterol.* Ann Pharmacother, 2022: p. 10600280221105169.
251. Ricci, A. and P. Ventura, *Givosiran for the treatment of acute hepatic porphyria.* Expert Rev Clin Pharmacol, 2022: p. 1-11.
252. Tschope, C. and A. Elsanhoury, *Treatment of Transthyretin Amyloid Cardiomyopathy: The Current Options, the Future, and the Challenges.* J Clin Med, 2022. **11**(8).
253. Betts, C., et al., *Pip6-PMO, A New Generation of Peptide-oligonucleotide Conjugates With Improved Cardiac Exon Skipping Activity for DMD Treatment.* Mol Ther Nucleic Acids, 2012. **1**: p. e38.
254. Therapeutics, S., *Sarepta Therapeutics Provides Update on SRP-5051 for the Treatment of Duchenne Muscular Dystrophy.* 2022.
255. Crooke, S.T., et al., *Antisense technology: A review.* Journal of Biological Chemistry, 2021. **296**: p. 100416.
256. Ferguson, M.W., et al., *Current and Possible Future Therapeutic Options for Huntington's Disease.* Journal of Central Nervous System Disease, 2022. **14**: p. 117957352210925.

257. Fire, A., et al., *Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans.* Nature, 1998. **391**(6669): p. 806-11.
258. Kim, D.H. and J.J. Rossi, *RNAi mechanisms and applications.* BioTechniques, 2008. **44**(5): p. 613-616.
259. Almeida, R. and R.C. Allshire, *RNA silencing and genome regulation.* Trends Cell Biol, 2005. **15**(5): p. 251-8.
260. Napoli, C., C. Lemieux, and R. Jorgensen, *Introduction of a Chimeric Chalcone Synthase Gene into Petunia Results in Reversible Co-Suppression of Homologous Genes in trans.* Plant Cell, 1990. **2**(4): p. 279-289.
261. Elbashir, S.M., et al., *Duplexes of 21-nucleotide RNAs mediate RNA interference in cultured mammalian cells.* Nature, 2001. **411**(6836): p. 494-498.
262. Neumeier, J. and G. Meister, *siRNA Specificity: RNAi Mechanisms and Strategies to Reduce Off-Target Effects.* Front Plant Sci, 2020. **11**: p. 526455.
263. Bagga, S., et al., *Regulation by let-7 and lin-4 miRNAs results in target mRNA degradation.* Cell, 2005. **122**(4): p. 553-63.
264. Zhang, X., V. Goel, and G.J. Robbie, *Pharmacokinetics of Patisiran, the First Approved RNA Interference Therapy in Patients With Hereditary Transthyretin-Mediated Amyloidosis.* J Clin Pharmacol, 2019.
265. Carvill, G.L. and H.C. Mefford, *Poison exons in neurodevelopment and disease.* Curr Opin Genet Dev, 2020. **65**: p. 98-102.
266. Scoles, D.R., E.V. Minikel, and S.M. Pulst, *Antisense oligonucleotides: A primer.* Neurol Genet, 2019. **5**(2): p. e323.
267. Lefebvre, S., et al., *Identification and characterization of a spinal muscular atrophy-determining gene.* Cell, 1995. **80**(1): p. 155-65.
268. Rochette, C.F., N. Gilbert, and L.R. Simard, *SMN gene duplication and the emergence of the SMN2 gene occurred in distinct hominids: SMN2 is unique to Homo sapiens.* Hum Genet, 2001. **108**(3): p. 255-66.
269. Monani, U.R., et al., *A single nucleotide difference that alters splicing patterns distinguishes the SMA gene SMN1 from the copy gene SMN2.* Hum Mol Genet, 1999. **8**(7): p. 1177-83.
270. Cartegni, L., et al., *Determinants of exon 7 splicing in the spinal muscular atrophy genes, SMN1 and SMN2.* Am J Hum Genet, 2006. **78**(1): p. 63-77.
271. Hastings, M.L., et al., *Tetracyclines that promote SMN2 exon 7 splicing as therapeutics for spinal muscular atrophy.* Sci Transl Med, 2009. **1**(5): p. 5ra12.
272. Prior, T.W., et al., *A positive modifier of spinal muscular atrophy in the SMN2 gene.* Am J Hum Genet, 2009. **85**(3): p. 408-13.
273. Passini, M.A., et al., *Antisense oligonucleotides delivered to the mouse CNS ameliorate symptoms of severe spinal muscular atrophy.* Sci Transl Med, 2011. **3**(72): p. 72ra18.
274. Hua, Y., et al., *Enhancement of SMN2 exon 7 inclusion by antisense oligonucleotides targeting the exon.* PLoS Biol, 2007. **5**(4): p. e73.
275. Hua, Y., et al., *Antisense masking of an hnRNP A1/A2 intronic splicing silencer corrects SMN2 splicing in transgenic mice.* Am J Hum Genet, 2008. **82**(4): p. 834-48.
276. Hoffman, E.P., R.H. Brown, Jr., and L.M. Kunkel, *Dystrophin: the protein product of the Duchenne muscular dystrophy locus.* Cell, 1987. **51**(6): p. 919-28.
277. Mendell, J.R., et al., *Eteplirsen for the treatment of Duchenne muscular dystrophy.* Ann Neurol, 2013. **74**(5): p. 637-47.

278. Lim, K.R., R. Maruyama, and T. Yokota, *Eteplirsen in the treatment of Duchenne muscular dystrophy.* Drug Des Devel Ther, 2017. **11**: p. 533-545.
279. Aartsma-Rus, A., et al., *Theoretic applicability of antisense-mediated exon skipping for Duchenne muscular dystrophy mutations.* Hum Mutat, 2009. **30**(3): p. 293-9.
280. Muntoni, F., S. Torelli, and A. Ferlini, *Dystrophin and mutations: one gene, several proteins, multiple phenotypes.* Lancet Neurol, 2003. **2**(12): p. 731-40.
281. Kole, R. and A.M. Krieg, *Exon skipping therapy for Duchenne muscular dystrophy.* Adv Drug Deliv Rev, 2015. **87**: p. 104-7.
282. Therapeutics, S., *Exondys 51 Identifying your child's mutation.* 2022.
283. Bushby, K.M. and D. Gardner-Medwin, *The clinical, genetic and dystrophin characteristics of Becker muscular dystrophy. I. Natural history.* J Neurol, 1993. **240**(2): p. 98-104.
284. Barp, A., et al., *Body composition and myokines in a cohort of patients with Becker muscular dystrophy.* Muscle & Nerve, 2022. **66**(1): p. 63-70.
285. Lim, K.H., et al., *Antisense oligonucleotide modulation of non-productive alternative splicing upregulates gene expression.* Nat Commun, 2020. **11**(1): p. 3501.
286. Han, Z., et al., *Antisense oligonucleotides increase Scn1a expression and reduce seizures and SUDEP incidence in a mouse model of Dravet syndrome.* Sci Transl Med, 2020. **12**(558).
287. Crunkhorn, S., *Treating Dravet syndrome with TANGO.* Nat Rev Drug Discov, 2020. **19**(10): p. 672.
288. Bass, B.L. and H. Weintraub, *An unwinding activity that covalently modifies its double-stranded RNA substrate.* Cell, 1988. **55**(6): p. 1089-98.
289. Wong, S.K., S. Sato, and D.W. Lazinski, *Substrate recognition by ADAR1 and ADAR2.* RNA, 2001. **7**(6): p. 846-58.
290. Savva, Y.A., L.E. Rieder, and R.A. Reenan, *The ADAR protein family.* Genome Biology, 2012. **13**(12): p. 252.
291. Consortium, G.T., *The Genotype-Tissue Expression (GTEx) project.* Nat Genet, 2013. **45**(6): p. 580-5.
292. Lamers, M.M., B.G. van den Hoogen, and B.L. Haagmans, *ADAR1: "Editor-in-Chief" of Cytoplasmic Innate Immunity.* Front Immunol, 2019. **10**: p. 1763.
293. Sun, T., et al., *Decoupling expression and editing preferences of ADAR1 p150 and p110 isoforms.* Proceedings of the National Academy of Sciences, 2021. **118**(12): p. e2021757118.
294. Piontkivska, H., et al., *ADAR Editing in Viruses: An Evolutionary Force to Reckon with.* Genome Biol Evol, 2021. **13**(11).
295. Samuel, C.E., *Adenosine deaminases acting on RNA (ADARs) are both antiviral and proviral.* Virology, 2011. **411**(2): p. 180-93.
296. Ringlander, J., et al., *Impact of ADAR-induced editing of minor viral RNA populations on replication and transmission of SARS-CoV-2.* Proceedings of the National Academy of Sciences, 2022. **119**(6): p. e2112663119.
297. Turunen, J. *AXIOMER® TECHNOLOGY Therapeutic oligonucleotides for directing site-specific A-to-I editing by endogenous ADAR enzymes*. [Presentation] 2017 January 14 2020]; Available from: https://www.proqr.com/wp-content/uploads/downloads/2017/11/Axiomer%20technology%20OTS%20170921.pdf.

298. Monian, P., et al., *Endogenous ADAR-mediated RNA editing in non-human primates using stereopure chemically modified oligonucleotides.* Nature Biotechnology, 2022.
299. Merkle, T., et al., *Precise RNA editing by recruiting endogenous ADARs with antisense oligonucleotides.* Nature Biotechnology, 2019. **37**(2): p. 133-138.
300. Qu, L., et al., *Programmable RNA editing by recruiting endogenous ADAR using engineered RNAs.* Nature Biotechnology, 2019. **37**(9): p. 1059-1069.
301. Gokirmak, T., et al., *Overcoming the challenges of tissue delivery for oligonucleotide therapeutics.* Trends Pharmacol Sci, 2021. **42**(7): p. 588-604.
302. Schlee, M. and G. Hartmann, *Discriminating self from non-self in nucleic acid sensing.* Nature Reviews Immunology, 2016. **16**(9): p. 566-580.
303. Gaidt, M.M., et al., *The DNA Inflammasome in Human Myeloid Cells Is Initiated by a STING-Cell Death Program Upstream of NLRP3.* Cell, 2017. **171**(5): p. 1110-1124.e18.
304. Pringle, I.A., et al., *Rapid identification of novel functional promoters for gene therapy.* J Mol Med (Berl), 2012. **90**(12): p. 1487-96.
305. Burel, S.A., et al., *Hepatotoxicity of high affinity gapmer antisense oligonucleotides is mediated by RNase H1 dependent promiscuous reduction of very long pre-mRNA transcripts.* Nucleic Acids Res, 2016. **44**(5): p. 2093-109.
306. Liang, X.H., et al., *Identification and characterization of intracellular proteins that bind oligonucleotides with phosphorothioate linkages.* Nucleic Acids Res, 2015. **43**(5): p. 2927-45.
307. Jack, D.B., *One hundred years of aspirin.* Lancet, 1997. **350**(9075): p. 437-9.
308. Jackson, A.L., et al., *Widespread siRNA "off-target" transcript silencing mediated by seed region sequence complementarity.* RNA, 2006. **12**(7): p. 1179-1187.
309. Naeem, M., et al., *Latest Developed Strategies to Minimize the Off-Target Effects in CRISPR-Cas-Mediated Genome Editing.* Cells, 2020. **9**(7): p. 1608.
310. Fedorov, Y., et al., *Off-target effects by siRNA can induce toxic phenotype.* RNA, 2006. **12**(7): p. 1188-96.
311. Jackson, A.L. and P.S. Linsley, *Recognizing and avoiding siRNA off-target effects for target identification and therapeutic application.* Nature Reviews Drug Discovery, 2010. **9**(1): p. 57-67.
312. Bartoszewski, R. and A.F. Sikorski, *Editorial focus: understanding off-target effects as the key to successful RNAi therapy.* Cellular & Molecular Biology Letters, 2019. **24**(1).
313. Hu, B., et al., *Therapeutic siRNA: state of the art.* Signal Transduction and Targeted Therapy, 2020. **5**(1).
314. Editorial, *Lasting impact of lipid nanoparticles.* Nature Reviews Materials, 2021. **6**(12): p. 1071-1071.
315. Mullard, A., *N-of-1 drugs push biopharma frontiers.* Nat Rev Drug Discov, 2020. **19**(3): p. 151-153.
316. Mole, S.E., *Batten disease: four genes and still counting.* Neurobiol Dis, 1998. **5**(5): p. 287-303.
317. Kim, J., et al., *Patient-Customized Oligonucleotide Therapy for a Rare Genetic Disease.* New England Journal of Medicine, 2019. **381**(17): p. 1644-1652.
318. Wethmar, K., J.J. Smink, and A. Leutz, *Upstream open reading frames: molecular switches in (patho)physiology.* Bioessays, 2010. **32**(10): p. 885-93.
319. Ingolia, N.T., et al., *Ribosome profiling reveals pervasive translation outside of annotated protein-coding genes.* Cell Rep, 2014. **8**(5): p. 1365-79.

320. Wang, D., et al., *A deep proteome and transcriptome abundance atlas of 29 healthy human tissues.* Mol Syst Biol, 2019. **15**(2): p. e8503.
321. Dworkin, M.J., *SHA-3 standard: Permutation-based hash and extendable-output functions.* 2015.
322. Diaz de Arce, A.J., W.L. Noderer, and C.L. Wang, *Complete motif analysis of sequence requirements for translation initiation at non-AUG start codons.* Nucleic Acids Res, 2018. **46**(2): p. 985-994.
323. McAlister, G.C., et al., *MultiNotch MS3 enables accurate, sensitive, and multiplexed detection of differential expression across cancer cell line proteomes.* Anal Chem, 2014. **86**(14): p. 7150-8.
324. Pomaznoy, M., B. Ha, and B. Peters, *GOnet: a tool for interactive Gene Ontology analysis.* BMC Bioinformatics, 2018. **19**(1).
325. Brar, G.A. and J.S. Weissman, *Ribosome profiling reveals the what, when, where and how of protein synthesis.* Nat Rev Mol Cell Biol, 2015. **16**(11): p. 651-64.
326. Steitz, J.A., *Polypeptide chain initiation: nucleotide sequences of the three ribosomal binding sites in bacteriophage R17 RNA.* Nature, 1969. **224**(5223): p. 957-64.
327. Wolin, S.L. and P. Walter, *Ribosome pausing and stacking during translation of a eukaryotic mRNA.* EMBO J, 1988. **7**(11): p. 3559-69.
328. Kiniry, S.J., A.M. Michel, and P.V. Baranov, *The GWIPS-viz Browser.* Curr Protoc Bioinformatics, 2018. **62**(1): p. e50.
329. Michel, A.M., et al., *GWIPS-viz: 2018 update.* Nucleic Acids Res, 2018. **46**(D1): p. D823-D830.
330. Ramirez, F., et al., *deepTools2: a next generation web server for deep-sequencing data analysis.* Nucleic Acids Res, 2016. **44**(W1): p. W160-5.
331. Paladin, L., et al., *The Feature Viewer: A visualization tool for positional annotations on a sequence.* Bioinformatics, 2020.
332. Abastado, J.P., et al., *Suppression of ribosomal reinitiation at upstream open reading frames in amino acid-starved cells forms the basis for GCN4 translational control.* Mol Cell Biol, 1991. **11**(1): p. 486-96.
333. Dmitriev, S.E., et al., *GTP-independent tRNA delivery to the ribosomal P-site by a novel eukaryotic translation factor.* J Biol Chem, 2010. **285**(35): p. 26779-26787.
334. Ivanov, I.P., et al., *Initiation context modulates autoregulation of eukaryotic translation initiation factor 1 (eIF1).* Proceedings of the National Academy of Sciences, 2010. **107**(42): p. 18056-18060.
335. Sample, P.J., et al., *Human 5 ´ UTR design and variant effect prediction from a massively parallel translation assay.* Nature Biotechnology, 2019. **37**(7): p. 803-809.
336. Tareen, A. and J.B. Kinney, *Logomaker: beautiful sequence logos in Python.* Bioinformatics, 2020. **36**(7): p. 2272-2274.
337. Nurk, S., et al., *The complete sequence of a human genome.* Science, 2022. **376**(6588): p. 44-53.
338. Adiconis, X., et al., *Comprehensive comparative analysis of 5'-end RNA-sequencing methods.* Nat Methods, 2018. **15**(7): p. 505-511.
339. McGlincy, N.J. and N.T. Ingolia, *Transcriptome-wide measurement of translation by ribosome profiling.* Methods, 2017. **126**: p. 112-129.
340. Ji, Z., et al., *Transcriptome-scale RNase-footprinting of RNA-protein complexes.* Nat Biotechnol, 2016. **34**(4): p. 410-3.

341. Kozak, M., *Constraints on reinitiation of translation in mammals.* Nucleic Acids Research, 2001. **29**(24): p. 5226-5232.
342. Kochetov, A.V., et al., *uORFs, reinitiation and alternative translation start sites in human mRNAs.* FEBS Letters, 2008. **582**(9): p. 1293-1297.
343. Karollus, A., Ž. Avsec, and J. Gagneur, *Predicting mean ribosome load for 5'UTR of any length using deep learning.* PLOS Computational Biology, 2021. **17**(5): p. e1008982.
344. Alfaro, J.A., et al., *The emerging landscape of single-molecule protein sequencing technologies.* Nature Methods, 2021. **18**(6): p. 604-617.
345. Dang, V.T., et al., *Identification of human haploinsufficient genes and their genomic proximity to segmental duplications.* Eur J Hum Genet, 2008. **16**(11): p. 1350-7.
346. Rehm, H.L., et al., *ClinGen — The Clinical Genome Resource.* New England Journal of Medicine, 2015. **372**(23): p. 2235-2242.
347. Huang, N., et al., *Characterising and predicting haploinsufficiency in the human genome.* PLoS Genet, 2010. **6**(10): p. e1001154.
348. Steinberg, J., et al., *Haploinsufficiency predictions without study bias.* Nucleic Acids Research, 2015. **43**(15): p. e101-e101.
349. Bathina, S. and U.N. Das, *Brain-derived neurotrophic factor and its clinical implications.* Archives of Medical Science, 2015. **6**: p. 1164-1178.
350. Wurster, C.D. and A.C. Ludolph, *Nusinersen for spinal muscular atrophy.* Therapeutic Advances in Neurological Disorders, 2018. **11**: p. 175628561875445.
351. Kowalski, P.S., et al., *Delivering the Messenger: Advances in Technologies for Therapeutic mRNA Delivery.* Mol Ther, 2019. **27**(4): p. 710-728.
352. Baden, L.R., et al., *Efficacy and Safety of the mRNA-1273 SARS-CoV-2 Vaccine.* N Engl J Med, 2021. **384**(5): p. 403-416.
353. Suay-Corredera, C., et al., *Protein haploinsufficiency drivers identify MYBPC3 variants that cause hypertrophic cardiomyopathy.* J Biol Chem, 2021. **297**(1): p. 100854.
354. Shen, Z.Q., et al., *CISD2 Haploinsufficiency Disrupts Calcium Homeostasis, Causes Nonalcoholic Fatty Liver Disease, and Promotes Hepatocellular Carcinoma.* Cell Rep, 2017. **21**(8): p. 2198-2211.
355. Liu, L., et al., *Mutation of the CDKN2A 5' UTR creates an aberrant initiation codon and predisposes to melanoma.* Nat Genet, 1999. **21**(1): p. 128-32.
356. Lace, B., et al., *Overview of Neuromuscular Disorder Molecular Diagnostic Experience for the Population of Latvia.* Neurology Genetics, 2022. **8**(3): p. e685.
357. Haddox, C.L., et al., *Phenotypic heterogeneity associated with germline GATA2 haploinsufficiency: a comprehensive kindred study.* Leuk Lymphoma, 2019. **60**(13): p. 3282-3286.
358. Oleaga-Quintas, C., et al., *Inherited GATA2 Deficiency Is Dominant by Haploinsufficiency and Displays Incomplete Clinical Penetrance.* J Clin Immunol, 2021. **41**(3): p. 639-657.
359. Huang, C.-H., et al., *Clinical spectrum and the comorbidities of Dravet syndrome in Taiwan and the possible molecular mechanisms.* Scientific Reports, 2021. **11**(1).
360. Crisafulli, S., et al., *Global epidemiology of Duchenne muscular dystrophy: an updated systematic review and meta-analysis.* Orphanet Journal of Rare Diseases, 2020. **15**(1).
361. Wang, R.T., et al., *DMD genotype correlations from the Duchenne Registry: Endogenous exon skipping is a factor in prolonged ambulation for individuals with a defined mutation subtype.* Human Mutation, 2018. **39**(9): p. 1193-1202.

362. Shen, Z.Q., Y.L. Huang, and T.F. Tsai, *Cisd2 haploinsufficiency: A driving force for hepatocellular carcinoma.* Mol Cell Oncol, 2018. **5**(3): p. e1441627.
363. Chen, Y.F., et al., *Cisd2 deficiency drives premature aging and causes mitochondria-mediated defects in mice.* Genes Dev, 2009. **23**(10): p. 1183-94.
364. Wu, C.Y., et al., *A persistent level of Cisd2 extends healthy lifespan and delays aging in mice.* Hum Mol Genet, 2012. **21**(18): p. 3956-68.
365. Amr, S., et al., *A homozygous mutation in a novel zinc-finger protein, ERIS, is responsible for Wolfram syndrome 2.* Am J Hum Genet, 2007. **81**(4): p. 673-83.
366. Harris, S.P., et al., *Hypertrophic Cardiomyopathy in Cardiac Myosin Binding Protein-C Knockout Mice.* Circulation Research, 2002. **90**(5): p. 594-601.
367. Glazier, A.A., A. Thompson, and S.M. Day, *Allelic imbalance and haploinsufficiency in MYBPC3-linked hypertrophic cardiomyopathy.* Pflugers Arch, 2019. **471**(5): p. 781-793.
368. Blake, D.J., J.M. Tinsley, and K.E. Davies, *Utrophin: A Structural and Functional Comparison to Dystrophin.* Brain Pathology, 1996. **6**(1): p. 37-47.
369. Speciale, A.A., et al., *Modelling Neuromuscular Diseases in the Age of Precision Medicine.* Journal of Personalized Medicine, 2020. **10**(4): p. 178.
370. Morrison, B.M., *Neuromuscular Diseases.* Semin Neurol, 2016. **36**(5): p. 409-418.
371. Müller, K.I., et al., *The prevalence of hereditary neuromuscular disorders in Northern Norway.* Brain and Behavior, 2021. **11**(1).
372. Soblechero-Martin, P., et al., *Utrophin modulator drugs as potential therapies for Duchenne and Becker muscular dystrophies.* Neuropathol Appl Neurobiol, 2021. **47**(6): p. 711-723.
373. Hardy, S., et al., *Magnesium-sensitive upstream ORF controls PRL phosphatase expression to mediate energy metabolism.* Proc Natl Acad Sci U S A, 2019. **116**(8): p. 2925-2934.
374. Si, X., et al., *Manipulating gene translation in plants by CRISPR-Cas9-mediated genome editing of upstream open reading frames.* Nat Protoc, 2020. **15**(2): p. 338-363.
375. Zhu, H., C. Li, and C. Gao, *Applications of CRISPR-Cas in agriculture and plant biotechnology.* Nat Rev Mol Cell Biol, 2020. **21**(11): p. 661-677.
376. Anzalone, A.V., et al., *Search-and-replace genome editing without double-strand breaks or donor DNA.* Nature, 2019. **576**(7785): p. 149-157.
377. Verdine, G.L. and L.D. Walensky, *The challenge of drugging undruggable targets in cancer: lessons learned from targeting BCL-2 family members.* Clin Cancer Res, 2007. **13**(24): p. 7264-70.
378. Beigel, J.H., et al., *Safety, tolerability, and pharmacokinetics of radavirsen (AVI-7100), an antisense oligonucleotide targeting influenza a M1/M2 translation.* Br J Clin Pharmacol, 2018. **84**(1): p. 25-34.
379. Ni, S., et al., *Recent Progress in Aptamer Discoveries and Modifications for Therapeutic Applications.* ACS Appl Mater Interfaces, 2021. **13**(8): p. 9500-9519.
380. Wahlestedt, C., *Natural antisense and noncoding RNA transcripts as potential drug targets.* Drug Discov Today, 2006. **11**(11-12): p. 503-8.
381. Li, L.C., et al., *Small dsRNAs induce transcriptional activation in human cells.* Proc Natl Acad Sci U S A, 2006. **103**(46): p. 17337-42.
382. Zhao, X., et al., *Mechanisms involved in the activation of C/EBPalpha by small activating RNA in hepatocellular carcinoma.* Oncogene, 2019. **38**(18): p. 3446-3457.
383. Huang, K.W., et al., *MTL-CEBPA Combined with Immunotherapy or RFA Enhances Immunological Anti-Tumor Response in Preclinical Models.* Int J Mol Sci, 2021. **22**(17).

384. Sarker, D., et al., *MTL-CEBPA, a Small Activating RNA Therapeutic Upregulating C/EBP-alpha, in Patients with Advanced Liver Cancer: A First-in-Human, Multicenter, Open-Label, Phase I Trial.* Clin Cancer Res, 2020. **26**(15): p. 3936-3946.
385. Zhou, J., et al., *Anti-inflammatory Activity of MTL-CEBPA, a Small Activating RNA Drug, in LPS-Stimulated Monocytes and Humanized Mice.* Mol Ther, 2019. **27**(5): p. 999-1016.
386. Setten, R.L., et al., *Development of MTL-CEBPA: Small Activating RNA Drug for Hepatocellular Carcinoma.* Curr Pharm Biotechnol, 2018. **19**(8): p. 611-621.
387. Yadav, R., et al., *Patient with Dravet syndrome: A case report.* Clin Case Rep, 2022. **10**(5): p. e05840.
388. Chilcott, E., et al., *Genetic therapeutic advancements for Dravet Syndrome.* Epilepsy Behav, 2022. **132**: p. 108741.
389. Bjurulf, B., et al., *Dravet syndrome in children-A population-based study.* Epilepsy Res, 2022. **182**: p. 106922.
390. Ding, J., et al., *SCN1A Mutation-Beyond Dravet Syndrome: A Systematic Review and Narrative Synthesis.* Front Neurol, 2021. **12**: p. 743726.
391. Wahlestedt, C., *Targeting long non-coding RNA to therapeutically upregulate gene expression.* Nat Rev Drug Discov, 2013. **12**(6): p. 433-46.
392. Modarresi, F., et al., *Inhibition of natural antisense transcripts in vivo results in gene-specific transcriptional upregulation.* Nat Biotechnol, 2012. **30**(5): p. 453-9.
393. Meng, L., et al., *Towards a therapy for Angelman syndrome by targeting a long non-coding RNA.* Nature, 2015. **518**(7539): p. 409-12.
394. Hsiao, J., et al., *Upregulation of Haploinsufficient Gene Expression in the Brain by Targeting a Long Non-coding RNA Improves Seizure Phenotype in a Model of Dravet Syndrome.* EBioMedicine, 2016. **9**: p. 257-277.
395. Crossley, B.M., et al., *Guidelines for Sanger sequencing and molecular assay monitoring.* J Vet Diagn Invest, 2020. **32**(6): p. 767-775.
396. Kluesner, M.G., et al., *EditR: A Method to Quantify Base Editing from Sanger Sequencing.* The CRISPR Journal, 2018. **1**(3): p. 239-250.
397. Montiel-Gonzalez, M.F., et al., *Correction of mutations within the cystic fibrosis transmembrane conductance regulator by site-directed RNA editing.* Proc Natl Acad Sci U S A, 2013. **110**(45): p. 18285-90.
398. Fukuda, M., et al., *Construction of a guide-RNA for site-directed RNA mutagenesis utilising intracellular A-to-I RNA editing.* Sci Rep, 2017. **7**: p. 41478.
399. Sinnamon, J.R., et al., *Site-directed RNA repair of endogenous Mecp2 RNA in neurons.* Proc Natl Acad Sci U S A, 2017. **114**(44): p. E9395-E9402.
400. Cox, D.B.T., et al., *RNA editing with CRISPR-Cas13.* Science, 2017. **358**(6366): p. 1019-1027.
401. Montiel-Gonzalez, M.F., I.C. Vallecillo-Viejo, and J.J. Rosenthal, *An efficient system for selectively altering genetic information within mRNAs.* Nucleic Acids Res, 2016. **44**(21): p. e157.
402. Vallecillo-Viejo, I.C., et al., *Abundant off-target edits from site-directed RNA editing can be reduced by nuclear localization of the editing enzyme.* RNA Biol, 2018. **15**(1): p. 104-114.
403. Katrekar, D., et al., *In vivo RNA editing of point mutations via RNA-guided adenosine deaminases.* Nat Methods, 2019. **16**(3): p. 239-242.

404. Katrekar, D., et al., *Efficient in vitro and in vivo RNA editing via recruitment of endogenous ADARs using circular guide RNAs.* Nat Biotechnol, 2022. **40**(6): p. 938-945.
405. Qu, L., et al., *Programmable RNA editing by recruiting endogenous ADAR using engineered RNAs.* Nat Biotechnol, 2019. **37**(9): p. 1059-1069.
406. Chen, C.X., et al., *A third member of the RNA-specific adenosine deaminase gene family, ADAR3, contains both single- and double-stranded RNA binding domains.* RNA, 2000. **6**(5): p. 755-67.
407. Oakes, E., P. Vadlamani, and H.A. Hundley, *Methods for the Detection of Adenosine-to-Inosine Editing Events in Cellular RNA.* Methods Mol Biol, 2017. **1648**: p. 103-127.
408. Merkle, T., et al., *Precise RNA editing by recruiting endogenous ADARs with antisense oligonucleotides.* Nat Biotechnol, 2019. **37**(2): p. 133-138.
409. ProQR, *Chemically modified single-stranded RNA-editing oligonucleotides.* United States Patent and Trademark Office, 2021.
410. Lee, D.S.M., et al., *Disrupting upstream translation in mRNAs is associated with human disease.* Nat Commun, 2021. **12**(1): p. 1515.
411. Winkelsas, A.M., et al., *Targeting the 5' untranslated region of SMN2 as a therapeutic strategy for spinal muscular atrophy.* Mol Ther Nucleic Acids, 2021. **23**: p. 731-742.
412. Sharma, S., et al., *APOBEC3A cytidine deaminase induces RNA editing in monocytes and macrophages.* Nat Commun, 2015. **6**: p. 6881.
413. Huang, X., et al., *Programmable C-to-U RNA editing using the human APOBEC3A deaminase.* EMBO J, 2020. **39**(22): p. e104741.
414. Davidson, N.O., et al., *Proposed nomenclature for the catalytic subunit of the mammalian apolipoprotein B mRNA editing enzyme: APOBEC-1.* RNA, 1995. **1**(1): p. 3.
415. Chen, S.H., et al., *Apolipoprotein B-48 is the product of a messenger RNA with an organ-specific in-frame stop codon.* Science, 1987. **238**(4825): p. 363-6.
416. Powell, L.M., et al., *A novel form of tissue-specific RNA processing produces apolipoprotein-B48 in intestine.* Cell, 1987. **50**(6): p. 831-40.
417. Ratcliff, J. and P. Simmonds, *Potential APOBEC-mediated RNA editing of the genomes of SARS-CoV-2 and other coronaviruses and its impact on their longer term evolution.* Virology, 2021. **556**: p. 62-72.
418. Lee, C.E., et al., *Rare Genetic Diseases: Nature's Experiments on Human Development.* iScience, 2020. **23**(5): p. 101123.
419. Prakash, V., M. Moore, and R.J. Yanez-Munoz, *Current Progress in Therapeutic Gene Editing for Monogenic Diseases.* Mol Ther, 2016. **24**(3): p. 465-74.
420. Smith, C.I.E., P. Bergman, and D.W. Hagey, *Estimating the number of diseases – the concept of rare, ultra-rare, and hyper-rare.* iScience, 2022. **25**(8): p. 104698.
421. Chen, B. and R.B. Altman, *Opportunities for developing therapies for rare genetic diseases: focus on gain-of-function and allostery.* Orphanet J Rare Dis, 2017. **12**(1): p. 61.
422. Marwaha, S., J.W. Knowles, and E.A. Ashley, *A guide for the diagnosis of rare and undiagnosed disease: beyond the exome.* Genome Med, 2022. **14**(1): p. 23.
423. Hedley, V., Murray, H; Rodwell, C; Aymé, S *Overview Report on the State of the Art of Rare Disease Activities in Europe.* RD Action, Data and policies for rare diseases, 2017.
424. Valcarcel-Nazco, C., et al., *Health-Related Quality of Life and Perceived Burden of Informal Caregivers of Patients with Rare Diseases in Selected European Countries.* Int J Environ Res Public Health, 2022. **19**(13).

425. Yang, G., et al., *The national economic burden of rare disease in the United States in 2019.* Orphanet J Rare Dis, 2022. **17**(1): p. 163.
426. Hillman, R.T., R.E. Green, and S.E. Brenner, *An unappreciated role for RNA surveillance.* Genome Biol, 2004. **5**(2): p. R8.
427. Brynczka, C.G., N, Okhorkova,O; Guenther,D; Kelkar,R; Yuva-Aydemir,Y;Smith,C;Al Abdullatif,A;Schwartz,B;Wahlestedt,C;Bumcrot,D;Sehgal,A, *Upregulation of SCN1A by regRNA-Targeting Antisense Oligonucleotide CMP-SCN.* DIA/FDA OLIGONUCLEOTIDE-BASED THERAPEUTICS CONFERENCE, 2022.
428. Nasevicius, A. and S.C. Ekker, *Effective targeted gene 'knockdown' in zebrafish.* Nat Genet, 2000. **26**(2): p. 216-20.
429. Bill, B.R., et al., *A primer for morpholino use in zebrafish.* Zebrafish, 2009. **6**(1): p. 69-77.
430. Arora, V., et al., *Transdermal use of phosphorodiamidate morpholino oligomer AVI-4472 inhibits cytochrome P450 3A2 activity in male rats.* Pharm Res, 2002. **19**(10): p. 1465-70.
431. Kipshidze, N.N., et al., *Intramural coronary delivery of advanced antisense oligonucleotides reduces neointimal formation in the porcine stent restenosis model.* J Am Coll Cardiol, 2002. **39**(10): p. 1686-91.
432. McCaffrey, A.P., et al., *A potent and specific morpholino antisense inhibitor of hepatitis C translation in mice.* Hepatology, 2003. **38**(2): p. 503-8.
433. Iversen, P.L., et al., *Efficacy of antisense morpholino oligomer targeted to c-myc in prostate cancer xenograft murine model and a Phase I safety study in humans.* Clin Cancer Res, 2003. **9**(7): p. 2510-9.
434. Devi, G.R., et al., *In vivo bioavailability and pharmacokinetics of a c-MYC antisense phosphorodiamidate morpholino oligomer, AVI-4126, in solid tumors.* Clin Cancer Res, 2005. **11**(10): p. 3930-8.
435. Stephens, Z.D., et al., *Big Data: Astronomical or Genomical?* PLoS Biol, 2015. **13**(7): p. e1002195.
436. Libbrecht, M.W. and W.S. Noble, *Machine learning applications in genetics and genomics.* Nat Rev Genet, 2015. **16**(6): p. 321-32.
437. Wagstaff, K., *Machine learning that matters.* arXiv preprint arXiv:1206.4656, 2012.
438. Leung, M.K., et al., *Machine learning in genomic medicine: a review of computational problems and data sets.* Proceedings of the IEEE, 2016. **104**(1): p. 176-197.
439. Chollet, F.o., *Deep learning with Python*. Second edition. ed. 2021, Shelter Island: Manning Publications. xxiii, 478 pages.

8 Appendix

The following attached Jupyter notebooks include the original code required to reproduce this work:

- uORFeome 1
- uORFeome 2
- BigWig File processor
- uORF Detective V2 DataPrep
- uORF Detective V2
- uORF Proteome Analysis
- uORF Transcriptome Analysis
- uORF Deep Learning

uORFeome_1

get_refseq_from_ucsc

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: Connects to UCSC, online access required.

This notebook uses the UCSC Genome Browser tables to get the most recent refseq protein coding mRNAs. It then uses the Biopython efetch wrapper to get the sequences and exon features for each mRNA. Duplicate copies mapped to both chrX and chrY are detected and exported. Any refseq ids not starting with NM_ are filtered out.

Output Files

Name	Type	Description
df_refseq_NM.pickle	Pickled DataFrame	Comprehensive Transcriptome ready for uORF analysis
chrX_chrY_duplicate_refseqs_same_id.csv	csv	66 refseqs with the same IDs that map on both X and Y chromosomes
df_refseq.bed	bed12	BED file generated for the human transcriptome
d_refseq_to_cds_start.pickle	Dictionary	Given refseq ID.v returns start of CDS coordinate on transcript
d_refseq_to_cds_end.pickle	Dictionary	Given refseq ID.v returns start of CDS coordinate on transcript
s_mane_select.pickle	Set	Given refseq ID.v returns start of CDS coordinate on transcript
d_mane_refseq_to_ensemble.pickle	Dictionary	Given refseq ID.v returns start of CDS coordinate on transcript
d_mane_refseq_to_coding.pickle	Dictionary	Given refseq ID.v returns start of CDS coordinate on transcript
d_mane_refseq_to_mane_status.pickle	Dictionary	Given refseq ID.v returns start of CDS coordinate on transcript

!

Required Libraries

In [ ]:

```
import numpy as np
import pandas as pd

pd.set_option("display.max_rows", 20)
import csv
import pickle

import pybedtools

### Biopython loads and details
from Bio import Entrez, SeqIO
from pybedtools import BedTool

api_key = (
    "d14717879c7708e9efefccce674a2510a008"
    # my API Key for NCBI allows 10 queries/sec
)
```

```
Entrez.api_key = api_key
    # If executing obtain youyr own API key otherwise
    #you are limited to 3 queries/sec
Entrez.email = "nenad.svrzikapa@paediatrics.ox.ac.uk"
```

1. Getting the RefSeq Genomic coordinates from UCSC

1.1 Load SQL Magic

In [ ]:
```
%load_ext sql
```

1.2 Connect to UCSC

In [ ]:
```
%sql mysql+pymysql://genome@genome-mysql.soe.ucsc.edu/hg38?charset=utf8&use_unicode=0
```

1.3 Retrieve RefSeq information from the ncbiRefSeqCurated table

This table contains version numbers.

In [ ]:
```
result = %sql SELECT name2 , name, chrom, strand, txStart, txEnd, cdsStart, cdsEnd, \
        exonCount,CONVERT(exonStarts USING utf8) AS exonStarts,\
        CONVERT(exonEnds USING utf8) AS exonEnds, \
        CONVERT(exonFrames USING utf8) AS exonFrames FROM ncbiRefSeqCurated;

df_refseq = result.DataFrame()
```

In [ ]:
```
df_refseq
```

1.4 Python byte string decoding to regular string

In [ ]:
```
b_df = df_refseq.select_dtypes([object])
b_df = b_df.stack().str.decode("utf-8").unstack()
for col in b_df:
    df_refseq[col] = b_df[col]
df_refseq
```

1.5 Filter and keep only refseq where nem column starts with "NM"

In [ ]:
```
df_refseq_NM = df_refseq[df_refseq["name"].str[0:2] == "NM"]
df_refseq_NM
```

1.6 Reset index

In [ ]:
```
df_refseq_NM.reset_index(inplace=True)
```

In [ ]:
```
df_refseq_NM = df_refseq_NM.drop("index", 1)
```

In [ ]:
```
df_refseq_NM
```

1.7 Filter only cannonical chromosome locations

In [ ]:
```
chr_set = {
    "chr1",
    "chr2",
    "chr3",
    "chr4",
    "chr5",
    "chr6",
    "chr7",
    "chr8",
    "chr9",
    "chr10",
    "chr11",
    "chr12",
    "chr13",
    "chr14",
    "chr15",
    "chr16",
    "chr17",
    "chr18",
    "chr19",
    "chr20",
    "chr21",
    "chr22",
    "chrX",
    "chrY",
}


df_refseq_NM = df_refseq_NM.loc[df_refseq_NM["chrom"].isin(chr_set)]
df_refseq_NM
```

In [ ]:
```
df_refseq_NM["name"].describe()
```

1.8 Filter out duplicated refseqs that map on X and Y chromosomes keep X only

There is a small number 66 refseqs with the same IDs that map on both X and Y chromosomes. This can cause overwrite errors down the line and I have only kept the X copy. The rest are output as a csv file named : chrX_chrY_duplicate_refseqs_same_id.csv

In [ ]:
```
df_refseq_NM = df_refseq_NM.assign(duplicated=df_refseq_NM["name"].duplicated())

df_refseq_NM.loc[df_refseq_NM["duplicated"] == True].sort_values(by=["name2"])

df_refseq_NM.loc[df_refseq_NM["duplicated"] == True].sort_values(by=["name2"]).to_csv(
    "chrX_chrY_duplicate_refseqs_same_id.csv"
)

df_filtered = df_refseq_NM[df_refseq_NM["duplicated"] == False]

len(df_filtered)
```

1.9 Drop temp columns index and duplicates

In [ ]:
```
df_filtered.drop(["duplicated"], axis=1, inplace=True)
df_filtered
```

In [ ]:
```
df_filtered["name"].describe()
```

In [ ]:
```
df_refseq_NM = df_filtered
```

2. Calculating new columns for BED12 generation

2.1 Generate block_sizes and block_starts columns

The Block sizes column represents the size of the exons and will be used to generate a BED12 temporary file. It can be calculated by subtracting the ExonStarts from the ExonEnds coordinates.

To do mathematical operations first the exonStarts and exonEnds need to be converted from string to list of integers.

After the sizes and starts are calculated, new columns are created to bedtools temporary object can be generated.

The bedtools object generated from each row is used to query the chromosome fasta files to get the refseq mrna fasta from the genomic coordinates as listed in the UCSC ncbirefseqcurated table.

In [ ]:
```
# convert exonStarts and exonEnds to integers
df_refseq_NM["exonStarts"] = df_refseq_NM["exonStarts"].apply(
    lambda x: [int(x) for x in x.split(",")[0:-1]]
)
df_refseq_NM["exonEnds"] = df_refseq_NM["exonEnds"].apply(
    lambda x: [int(x) for x in x.split(",")[0:-1]]
)
```

In [ ]:
```
def get_block_sizes(e_Ends, e_Starts):
    """
    Subtracts values in two lists store in pandas df columns
    returns a list
    """
    temp = []
    for i, j in zip(e_Ends, e_Starts):
        temp.append(i - j)
    return temp


def get_block_starts(txStart, e_Starts):
    """
    Subtracts integer from every value in list in pandas df column
    """
    temp = []
    for i in e_Starts:
        temp.append(i - txStart)
    return temp
```

In [ ]:
```
df_refseq_NM["block_sizes"] = df_refseq_NM.apply(
    lambda x: get_block_sizes(x.exonEnds, x.exonStarts), axis=1
)
df_refseq_NM["blockStarts"] = df_refseq_NM.apply(
    lambda x: get_block_starts(x.txStart, x.exonStarts), axis=1
)
df_refseq_NM.head()
```

2.2 Adding the score and color default values for BED12

In [ ]:
```
# default values for score and color
```

```
df_refseq_NM["score"] = 1000
df_refseq_NM["color"] = ["241,174,28"] * len(df_refseq_NM)
```

2.3 Remap the list containing columns to strings to make a valid BED12 object

Creates str_block_sizes, str_exonStarts

In [ ]:
```
df_refseq_NM["str_block_sizes"] = [
    ",".join(map(str, l)) for l in df_refseq_NM["block_sizes"]
]
df_refseq_NM["str_blockStarts"] = [
    ",".join(map(str, l)) for l in df_refseq_NM["blockStarts"]
]
```

In [ ]:
```
df_refseq_NM.head(50)
```

In [ ]:
```
df_test = df_refseq_NM.loc[df_refseq_NM.name == "NM_001330306.2"][
    [
        "chrom",
        "txStart",
        "txEnd",
        "name",
        "score",
        "strand",
        "cdsStart",
        "cdsEnd",
        "color",
        "exonCount",
        "str_block_sizes",
        "str_blockStarts",
    ]
]
df_test
```

2.4 Outputof a BED12 file of the refseq mRNA NM_ database

In [ ]:
```
df_refseq_bed12 = df_refseq_NM[
    [
        "chrom",
        "txStart",
        "txEnd",
        "name",
        "score",
        "strand",
        "cdsStart",
        "cdsEnd",
        "color",
        "exonCount",
        "str_block_sizes",
        "str_blockStarts",
    ]
]
df_refseq_bed12.to_csv(
    "../references/refseq/bed/df_refseq.bed",
    sep="\t",
    index=False,
    header=False,
    quoting=csv.QUOTE_NONE,
)
```

3. Using generated BED12 to get the mRNA sequences with pybedtools

In [ ]:

```
def get_fasta(
    chrom,
    txStart,
    txEnd,
    name,
    score,
    strand,
    cdsStart,
    cdsEnd,
    color,
    exonCount,
    str_block_sizes,
    str_blockStarts,
):
    """
    Provided columns that make up a BED12 format it returns a string
    for each row of the mRNA sequence
    """

    d = {
        "chrom": chrom,
        "txStart": txStart,
        "txEnd": txEnd,
        "name": name,
        "score": score,
        "strand": strand,
        "cdsStart": cdsStart,
        "cdsEnd": cdsEnd,
        "color": color,
        "exonCount": exonCount,
        "str_block_sizes": str_block_sizes,
        "str_blockStarts": str_blockStarts,
    }
    # note that you must pass an index if scalar values are passed
    df = pd.DataFrame(d, index=[0])
    # a = BedTool("df_test.bed")
    # create a BedTool object from the dataframe row
    x = pybedtools.BedTool.from_dataframe(df)
    fasta = BedTool("../references/hg38/" + d["chrom"] + ".fa")
    x = x.sequence(fi=fasta, s=True, split=True)
    return open(x.seqfn).read().split(")")[1].upper()


df_refseq_NM["sequence"] = df_refseq_NM.apply(
    lambda x: get_fasta(
        x.chrom,
        x.txStart,
        x.txEnd,
        x.name,
        x.score,
        x.strand,
        x.cdsStart,
        x.cdsEnd,
        x.color,
        x.exonCount,
        x.str_block_sizes,
        x.str_blockStarts,
    ),
    axis=1,
)
```

In [ ]:
```
df_refseq_NM["sequence"] = df_refseq_NM["sequence"].replace(r"\n", "", regex=True)
df_refseq_NM.head()
```

In [ ]:
```
df_refseq_NM.head(20)
```

4. Quality Control

Biopython was used to get the refseq sequences from NCBI. This method is much faster, but for the purpose of working with the most current information at UCSC, we get the sequences from the genomic coordinates. However,the dictionary created with refseq_fna_df_dictionary.ipynb which uses the refseq ID with version as key and returns sequence as value can be used to verify the accuracy of querying the sequences with their genomic coordinates.

Sample with smaller samples and check the lengths. The refseq mrna fna sequences tend to have longer Poly-A tails which don't map to the genome. The genome BED12 fasta acquisition was successful.

4.1 Load the pickled d_refseq_fna_fasta dictionary

In [ ]:
```
# Load the d_refseq_fna_fasta dictionary
with open(
    "../references/refseq/dictionaries/d_refseq_fna_fasta.pickle", "rb"
) as handle:
    d_refseq_fna_fasta = pickle.load(handle)
```

4.2 Randomly sample rows from the df_refseq_NM dataframe

It was found that sequences may differ in the d_refseq_fna_fasta.pickle in the length of the poly-A tails

In [ ]:
```
df_refseq_NM.rename({"name2": "gene_name", "name": "refseq.v"}, axis=1, inplace=True)

df_sample = df_refseq_NM.sample(n=20)[["refseq.v", "strand", "sequence"]]
df_sample["d_sequence"] = df_sample["refseq.v"].map(d_refseq_fna_fasta)
df_sample["len_sequence"] = df_sample["sequence"].apply(lambda x: len(x))
df_sample["lean_d_sequence"] = df_sample["d_sequence"].apply(lambda x: len(x))
```

In [ ]:
```
df_sample
```

In [ ]:
```
df_sample["sequence"].equals(df_sample["d_sequence"])
```

5. Add CDS coordinates on RNA level

Note that the genome is 1 indexed and as such the start coordinates that NCBI adds have +1. As such for pythonic operations it is prefered that this is brought back to 0 indexed. The genomic coordinates in their tables are 1 indexed and require no change because they query a 1 indexed genome.

5.1 Get the id and cds columns from the UCSC ncbiRefSeqCds table

In [ ]:
```
result_cds = %sql SELECT id , cds  FROM ncbiRefSeqCds;
```

In [ ]:
```
df_cds = result_cds.DataFrame()
```

5.2 Byte strings to string conversion

In [ ]:
```
b_df_cds = df_cds.select_dtypes([object])
b_df_cds = b_df_cds.stack().str.decode("utf-8").unstack()
for col in b_df_cds:
    df_cds[col] = b_df_cds[col]
df_cds.head()
```

5.3 Keep only refseqs starting with NM_

In [ ]:
```
df_cds = df_cds[df_cds["id"].str[0:2] == "NM"]
```

5.4 Filter out any incomplete cds information and split start and end in separate columns

In [ ]:
```
df_cds["cds_complete"] = df_cds["cds"].map(lambda x: True if ".." in x else False)
df_cds.drop(df_cds[df_cds["cds_complete"] == False].index, inplace=True)
df_cds["cds_start"] = df_cds["cds"].apply(lambda x: x.split(".")[0])
df_cds["cds_end"] = df_cds["cds"].apply(lambda x: x.split(".")[2])
```

5.5 Create and pickle the refseq if to cds_start or cds_end dictionaries

These dictionaries are used to map the CDS start and end values to the df_refseq_NM dataframe

In [ ]:
```
d_refseq_to_cds_start = dict(zip(df_cds.id, df_cds.cds_start))
d_refseq_to_cds_end = dict(zip(df_cds.id, df_cds.cds_end))

with open(
    "../references/refseq/dictionaries/d_refseq_to_cds_start.pickle", "wb"
) as handle:
    pickle.dump(d_refseq_to_cds_start, handle, protocol=pickle.HIGHEST_PROTOCOL)
with open(
    "../references/refseq/dictionaries/d_refseq_to_cds_end.pickle", "wb"
) as handle:
    pickle.dump(d_refseq_to_cds_end, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

5.6 Map the CDS coordinates to new columns in the df_refseq_NM dataframe

In [ ]:
```
df_refseq_NM["cds_start"] = df_refseq_NM["refseq.v"].map(d_refseq_to_cds_start)
df_refseq_NM["cds_end"] = df_refseq_NM["refseq.v"].map(d_refseq_to_cds_end)
```

5.7 Remove anything that can not be converted to an int, convert columns to int, subtract 1 from cds_start

In [ ]:
```
df_refseq_NM = df_refseq_NM[df_refseq_NM.cds_start.astype(str).str.isdigit()]
df_refseq_NM = df_refseq_NM[df_refseq_NM.cds_end.astype(str).str.isdigit()]

df_refseq_NM["cds_start"] = df_refseq_NM["cds_start"].astype(int)
df_refseq_NM["cds_end"] = df_refseq_NM["cds_end"].astype(int)

df_refseq_NM["cds_start"] = df_refseq_NM["cds_start"] - 1
```

6. Get the MANE + SELECT and MANE + matched Ensemble IDs

MANE stands for Matched Annotation of NCBI and Ensemble, and represents the most expressed and most representative isoform. SELECT includes MANE, but also has refseqs which are not matched with

ENSEMBLE.

6.1 MANE+SELECT is queried from the ncbiRefSeqSelect at UCSC

This is followed by byte string conversion

In [ ]:
```
result = %sql SELECT name FROM ncbiRefSeqSelect;
df_mane_select = result.DataFrame()
b_df_mane_select = df_mane_select.select_dtypes([object])
b_df_mane_select = b_df_mane_select.stack().str.decode('utf-8').unstack()
for col in b_df_mane_select:
    df_mane_select[col] = b_df_mane_select[col]
```

6.2 Create a set of the refseq IDs from the MANE+SELECT collection and pickled it

In [ ]:
```
s_mane_select = set(df_mane_select["name"])

with open("../references/refseq/sets/s_mane_select.pickle", "wb") as handle:
    pickle.dump(s_mane_select, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

6.3 MANE is processed from the bigbed binary file at this URL

https://genome.ucsc.edu/cgi-bin/hgTables?db=hg38&hgta_group=genes&hgta_track=mane&hgta_table=mane&hgta_doSchema=describe+table+schema

using the UCSC bigbedtobed conversion tool. It is then loaded using pybedtools to a dataframe and the appropriate columns are selected.

Here we create 3 dictionaries:

1. refseq -> ensemble
2. refseq -> coding (is the refseq protein coding)
3. refseq -> MANE status (clinical, select)

and one set: s_mane - a set of the refseqs.

In [ ]:
```
mane_bed = BedTool("../references/refseq/MANE/mane.bed")
df_mane = mane_bed.to_dataframe(header=None)
df_mane = df_mane[[21, 3, 19, 24]]
d_mane_refseq_to_ensemble = dict(zip(df_mane[21], df_mane[3]))
d_mane_refseq_to_coding = dict(zip(df_mane[21], df_mane[19]))
d_mane_refseq_to_mane_status = dict(zip(df_mane[21], df_mane[24]))
s_mane = set(df_mane[21])

with open(
    "../references/refseq/dictionaries/d_mane_refseq_to_ensemble.pickle", "wb"
) as handle:
    pickle.dump(d_mane_refseq_to_ensemble, handle, protocol=pickle.HIGHEST_PROTOCOL)

with open(
    "../references/refseq/dictionaries/d_mane_refseq_to_coding.pickle", "wb"
) as handle:
    pickle.dump(d_mane_refseq_to_coding, handle, protocol=pickle.HIGHEST_PROTOCOL)

with open(
    "../references/refseq/dictionaries/d_mane_refseq_to_mane_status.pickle", "wb"
) as handle:
```

```
    pickle.dump(d_mane_refseq_to_mane_status, handle, protocol=pickle.HIGHEST_PROTOCOL)

with open("../references/refseq/sets/s_mane.pickle", "wb") as handle:
    pickle.dump(s_mane, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

In [ ]:
```
df_mane.head(3)
```

6.4 Mapping the mane_select, mane, and ensemble columns to df_refseq_NM

In [ ]:
```
df_refseq_NM["ensemble"] = df_refseq_NM["refseq.v"].map(d_mane_refseq_to_ensemble)
df_refseq_NM["mane_select"] = df_refseq_NM["refseq.v"].apply(
    lambda x: True if x in s_mane_select else False
)
df_refseq_NM["mane"] = df_refseq_NM["refseq.v"].apply(
    lambda x: True if x in s_mane else False
)
df_refseq_NM["type"] = df_refseq_NM["refseq.v"].map(d_mane_refseq_to_mane_status)
df_refseq_NM["coding"] = df_refseq_NM["refseq.v"].map(d_mane_refseq_to_coding)
```

In [ ]:
```
df_refseq_NM["refseq"] = df_refseq_NM["refseq.v"].apply(lambda x: x.split(".")[0])
```

In [ ]:
```
column_names = [
    "gene_name",
    "refseq.v",
    "refseq",
    "ensemble",
    "mane_select",
    "mane",
    "chrom",
    "strand",
    "txStart",
    "txEnd",
    "cdsStart",
    "cdsEnd",
    "exonCount",
    "exonStarts",
    "exonEnds",
    "exonFrames",
    "duplicated",
    "block_sizes",
    "score",
    "color",
    "str_block_sizes",
    "str_exonStarts",
    "blockStarts",
    "str_blockStarts",
    "cds_start",
    "cds_end",
    "type",
    "coding",
    "sequence",
]
df_refseq_NM = df_refseq_NM.reindex(columns=column_names)
df_refseq_NM.head(3)
```

In [ ]:
```
with open("../references/refseq/dataframes/df_refseq_NM.pickle", "wb") as handle:
    pickle.dump(df_refseq_NM, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

uORFeome_2

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: Output from get_refseqs_from_ucsc.ipynb

This notebook uses the processed df_refseq_NM dataframe in order to calculate and organize various features related to upstream open reading frames (uORFs)

Required Libraries

In [ ]:
```
import pickle
import re

import numpy as np
import pandas as pd

pd.set_option("display.max_columns", None)
import glob
import hashlib
import math

import pyBigWig
```

1. Loading of the starting processed dataframe

1.1 Open the df_refseq_NM.pickle starting dataframe

The dataframe is generated and pickled by the get_refseqs_from_ucsc.ipynb The get_refseqs_from_ucsc.ipynb queries the latest version of the ncbiRefSeqCurated table from UCSC. To work with the most recent version of the refseq database in UCSC the df_refseq_NM.pickle file can be regenerated by re-running the get_refseqs_from_ucsc.ipynb analysis.

In [ ]:
```
# Load the d_refseq_fna_fasta dictionary
with open("../references/refseq/dataframes/df_refseq_NM.pickle", "rb") as handle:
    df_uORF = pickle.load(handle)
```

1.2 Load and open the BigWig Files used for querying Initiations, Ribosome Footprints, mRNA, Custom mRNA

In [ ]:
```
init_file = glob.glob("../references/bigwigs/twips_initiation/*.bw")
bw_init = pyBigWig.open(init_file[0])

footprints_file = glob.glob("../references/bigwigs/twips_footprints/*.bw")
bw_footprints = pyBigWig.open(footprints_file[0])

mrna_file = glob.glob("../references/bigwigs/twips_mrna/*.bw")
bw_mrna = pyBigWig.open(mrna_file[0])

custom_file = glob.glob("../references/bigwigs/twips_custom/*.bw")
bw_custom = pyBigWig.open(custom_file[0])
```

2. Helper functions and dictionaries

Name	Type	Description
d_aa	Dictionary	Codon to Amino Acid one-letter translation dictionary
translate_to_aa	Function	Given a DNA sequence translates to protein sequence
get_reading_frame_mrna	Function	Given coordinate and strand returns the reading frame relative to start
get_reading_frame	Function	Given CDS and uORF start coordinates returns the reading frame of uORF relative to CDS
get_exon_coordinate_list	Function	Given t_exon and strand returns a list of hg38 coordinates for a given transcript. Used to query bigwigs.
d_stop_codon	Dictionary	Stop Codon to Stop codon, Stop codon name
l_stop_codons	List	List of stop codons
get_uORFs	Function	Given 5'UTR coordinates, strand,l_exon_coordinates,chromosome,start codon,returns (uORF transcript order rank,uORF transcript coordinates, uORF genomic coordinates, uORF sequence, uORF translation, uORF start initiation, uORF stop codon, uORF frame relative to CDS, uORF kozak type, tuple of unique uORF ID made of uORF start and uORF sequence.
get_bw_value	Function	Given chromosome, coordinate, bigwif file, returns the value at that coordinate or 0 if the value is NaN

In [ ]:

```
# translation dictionary
d_aa = {
    "GCT": "A",
    "GCC": "A",
    "GCA": "A",
    "GCG": "A",
    "CGT": "R",
    "CGC": "R",
    "CGA": "R",
    "CGG": "R",
    "AGA": "R",
    "AGG": "R",
    "AAT": "N",
    "AAC": "N",
    "GAT": "D",
    "GAC": "D",
    "TGT": "C",
    "TGC": "C",
    "CAA": "Q",
    "CAG": "Q",
    "GAA": "E",
    "GAG": "E",
    "GGT": "G",
    "GGC": "G",
    "GGA": "G",
    "GGG": "G",
    "CAT": "H",
    "CAC": "H",
    "ATT": "I",
    "ATC": "I",
    "ATA": "I",
    "CTT": "L",
    "CTC": "L",
    "CTA": "L",
    "CTG": "L",
    "TTA": "L",
    "TTG": "L",
    "AAA": "K",
    "AAG": "K",
    "TTT": "F",
```

```
    "TTC": "F",
    "CCT": "P",
    "CCC": "P",
    "CCA": "P",
    "CCG": "P",
    "TCT": "S",
    "TCC": "S",
    "TCA": "S",
    "TCG": "S",
    "AGT": "S",
    "AGC": "S",
    "ACT": "T",
    "ACC": "T",
    "ACA": "T",
    "ACG": "T",
    "TGG": "W",
    "TAT": "Y",
    "TAC": "Y",
    "GTT": "V",
    "GTC": "V",
    "GTA": "V",
    "GTG": "V",
    "TAG": "",
    "TAA": "",
    "TGA": "",
    "ATG": "M",
}


def translate_to_aa(s, d):
    """
    Takes string in ATGC format and a translation dictionary to return amino acid
    sequence
    """
    i = 0
    l = len(s)
    result = ""
    for i in range(0, l, 3):
        result += d[s[i : i + 3]]
    return result


def get_reading_frame_mrna(coordinate, strand):
    """
    Given a coordinate returns the reading frame based on mRNA start
    """
    frame = 0
    if (coordinate) % 3 == 0:
        frame = 1
    elif (coordinate) % 3 == 1:
        frame = 2
    else:
        frame = 3

    if strand == "-":
        frame = -frame
    return frame


def get_reading_frame(coordinate_uORF, coordinate_CDS):
    """
    Given a coordinates of CDS and uORF,
    Returns the reading frame of uORF relative to uORF
    Options are 0,1,2 where 1 is -1 and 2 is -2, while 0 is the same reading frame as
    the CDS.
    """
```

```
    return (coordinate_CDS - coordinate_uORF) % 3


def get_exon_coordinate_list(t_exons, strand):
    """
    Calculates and returns a list of all exon coordinates.
    These coordinates are on hg38 level and can be used to access the bigwig information.
    They can also be used to map everythin from mRNA on genomic level.
    These coordinates are also used to generate a bigbed track of all uORFs.
    """
    exon_ranges = list(t_exons)
    l_exon_coordinates = []

    for rng in exon_ranges:
        if strand == "-":
            l_exon_coordinates.append(np.arange(rng[0] + 1, rng[1] + 1)[::-1])
        else:
            l_exon_coordinates.append(np.arange(rng[0], rng[1]))

    return [coordinate for exon in l_exon_coordinates for coordinate in exon]


d_stop_codon = {
    "TAG": ("TAG", "Amber"),
    "TGA": ("TGA", "Opal"),
    "TAA": ("TAA", "Ochre"),
}
l_stop_codons = ["TAG", "TGA", "TAA"]


def get_uORFs(
    sequence,
    cds_start,
    strand,
    l_exon_coordinates,
    chrom,
    gene_name,
    refseq_v,
    init_codon="ATG",
):
    """
    returns 0 based coordinates of uORF positions
    """
    var_seq = sequence
    mrna_cds_start = cds_start
    strand = strand
    gene_name = gene_name
    refseq_v = refseq_v
    # Evaluating lists of Kozak
    l_strong_kozak = [
        m.start() + 3
        for m in re.finditer("[AG].." + init_codon + "G", var_seq[0:mrna_cds_start])
    ]
    l_adequate_5_kozak = [
        m.start() + 3
        for m in re.finditer("[AG].." + init_codon + "[ACT]", var_seq[0:mrna_cds_start])
    ]
    l_adequate_3_kozak = [
        m.start() + 3
        for m in re.finditer("[CT].." + init_codon + "G", var_seq[0:mrna_cds_start])
    ]

    # find coordinates of all initiation codons in 5'UTR
    utr_ATG_coordinates = [
        m.start() for m in re.finditer(init_codon, var_seq[0:mrna_cds_start])
    ]
```

```
# pORF_init and mRNA
pORF_init = get_bw_value(chrom, l_exon_coordinates[mrna_cds_start], bw_init)
pORF_mrna = get_bw_value(chrom, l_exon_coordinates[mrna_cds_start], bw_mrna)
pORF_footprints = get_bw_value(
    chrom, l_exon_coordinates[mrna_cds_start], bw_footprints
)

for x, start in enumerate(utr_ATG_coordinates):
    kozak = ""
    for i in range(start, len(var_seq) - 2, 3):
        evaluate = var_seq[i : i + 3]
        if evaluate in l_stop_codons:
            # print (x,start+1,(i+3))
            if start in l_strong_kozak:
                kozak = ("S", sequence[start - 3 : start + 4])
            if start in l_adequate_5_kozak:
                kozak = ("A", sequence[start - 3 : start + 4])
            if start in l_adequate_3_kozak:
                kozak = ("A", sequence[start - 3 : start + 4])
            else:
                kozak = ("W", sequence[start - 3 : start + 4])

            frame = get_reading_frame(start, cds_start)
            # Calculating the uORF initiation, footprint, mrna, pORF is outside
            # the loop
            uORF_init = get_bw_value(chrom, l_exon_coordinates[start], bw_init)
            uORF_footprints = get_bw_value(
                chrom, l_exon_coordinates[start], bw_footprints
            )
            uORF_mrna = get_bw_value(chrom, l_exon_coordinates[start], bw_mrna)

            # taking the ratio of initiation and footprint normalized to mRNA for
            # each uORF
            ratio_init = get_uORF_pORF_ratio(
                uORF_init, uORF_mrna, pORF_init, pORF_mrna
            )
            ratio_footprints = get_uORF_pORF_ratio(
                uORF_footprints, uORF_mrna, pORF_footprints, pORF_mrna
            )

            minimal_uORF = False

            uORF_seq = sequence[start : i + 3]
            if uORF_seq in ["ATGTAG", "ATGTGA", "ATGTAA"]:
                minimal_uORF = True
            g_uorf_start = l_exon_coordinates[start]
            g_uorf_stop = l_exon_coordinates[i + 2]
            tiss_span = False
            if i + 3 > mrna_cds_start:
                tiss_span = True
            utr_ATG_coordinates[x] = (
                (
                    hashlib.shake_256(my_string.encode()).hexdigest(
                        3
                    ),  # unique uORF ID
                    str(chrom)
                    + "_"
                    + str(g_uorf_start)
                    + "_"
                    + str(len(uORF_seq)),  # uORF genomic ID 0
                    x + 1,  # Transcript Rank 1
                    (start, i + 2),  # Transcript coordinates 2
                    round(
                        (start / mrna_cds_start) * 100, 1
                    ),  # Percent position from CDS
```

```
                            mrna_cds_start - start,  # distance to CDS
                            (g_uorf_start, g_uorf_stop),  # Genomic coordinates 2
                            uORF_seq,  # uORF mRNA sequence 3
                            minimal_uORF,  # is the uORF minimal or not 4
                            translate_to_aa(uORF_seq, d_aa),  # uORF Translation 5
                            (
                                uORF_init,
                                uORF_footprints,
                                uORF_mrna,
                            ),  # uORF initiation,footprints,mrna 6
                            (
                                pORF_init,
                                pORF_footprints,
                                pORF_mrna,
                            ),  # pORF initiation,footprints,mrna 7
                            ratio_init,  # the ratio of pORF vs uORF init normalized to mRNA
                            ratio_footprints,  # pORF/uORF footprint ratio normalized to mRNA
                            round(
                                start / mrna_cds_start * 100, 1
                            ),  # uORF %position 5'UTR 10
                            tiss_span,  # TISS span yes or no 11
                            d_stop_codon[evaluate],  # stop codon type 12
                            frame,  # frame relative to pORF start 13
                            kozak,  # kozak type and sequence 14
                            (chrom, g_uorf_start, uORF_seq),
                        ),
                        15,
                    )
                    break

    # print (utr_ATG_coordinates)
    if not utr_ATG_coordinates:
        return np.nan
    else:
        return utr_ATG_coordinates


def f(x):
    """
    A Function used to combine values of 2 lists residing in two columns,
    into a new column containing zipped values.
    """
    temp = []
    for i, j in zip(x["t_exonStarts"], x["t_exonEnds"]):
        temp.append([i, j])
    return temp


def get_bw_value(chromosome, coordinate, bw_file):
    """
    Given chromosome, coordinate, bigwig file type
    Returns the value at that coordinate from the bigwig file.
    Returns 0 for values that are NaN
    """
    bw = bw_file
    if math.isnan(bw.values(chromosome, coordinate - 1, coordinate)[0]):
        result = 0
    else:
        result = int(bw.values(chromosome, coordinate - 1, coordinate)[0])
    return result


def get_uORF_pORF_ratio(uORF_coordinate, uORF_mrna, pORF_coordinate, pORF_mrna):
    """
    Calculates the ratio of initiation, or footprints to mrna relative to pORF coordinate.
    Returns 1 number.
```

```
    0 values are converted to 1 to avoid division by 0.
    """

    if uORF_coordinate == 0:
        uORF_coordinate = 1
    if pORF_coordinate == 0:
        pORF_coordinate = 1
    if uORF_mrna == 0:
        uORF_mrna = 1
    if pORF_mrna == 0:
        pORF_mrna = 1

    return round(
        ((uORF_coordinate / uORF_mrna) / (pORF_coordinate / pORF_mrna)) * 100, 1
    )
```

In [ ]:
```
assert (
    len(set(d_aa.keys()))
) == 64  # assertion for uniquiness of Amino acid codons and length of 64
```

3. Analysis

3.1 Drop all rows where the CDS start is the same as the Transcript Start

These rows have no 5'UTRs

In [ ]:
```
df_uORF.drop(df_uORF[df_uORF["txStart"] == df_uORF["cdsStart"]].index, inplace=True)
```

3.2 Strand-aware reorganization of coordinates:

1. Transcript Start and End coordinates
2. Transcript cdsStart and cdsEnd coordinates
3. Transcript Exon starts and exon ends coordinates

In [ ]:
```
df_uORF["t_txStart"] = df_uORF.apply(
    lambda x: x["txStart"] if x["strand"] == "+" else x["txEnd"], axis=1
)
df_uORF["t_txEnd"] = df_uORF.apply(
    lambda x: x["txEnd"] if x["strand"] == "+" else x["txStart"], axis=1
)
df_uORF["t_cdsStart"] = df_uORF.apply(
    lambda x: x["cdsStart"] if x["strand"] == "+" else x["cdsEnd"], axis=1
)
df_uORF["t_cdsEnd"] = df_uORF.apply(
    lambda x: x["cdsEnd"] if x["strand"] == "+" else x["cdsStart"], axis=1
)

df_uORF["t_exonStarts"] = df_uORF.apply(
    lambda x: x["exonStarts"]
    if x["strand"] == "+"
    else sorted(x["exonStarts"], reverse=True),
    axis=1,
)
df_uORF["t_exonEnds"] = df_uORF.apply(
    lambda x: x["exonEnds"]
    if x["strand"] == "+"
    else sorted(x["exonEnds"], reverse=True),
```

```
    axis=1,
)
```

3.3 Combining the linked t_exonStarts and t_exonEnds into a single t_exons column

In [ ]:
```
df_uORF["t_exons"] = df_uORF.apply(f, axis=1)
```

In [ ]:
```
df_uORF.loc[df_uORF["refseq.v"] == "NM_005523.6"]
```

3.4 Calling the get_uORF function to map all uORFs for AUG starts and near-cognate initiation bases

In [ ]:
```
print("Processing AUG")
df_uORF["AUG_uORFs"] = df_uORF.apply(
    lambda x: get_uORFs(
        x.sequence,
        x.cds_start,
        x.strand,
        x.l_exon_coordinates,
        x.chrom,
        x.gene_name,
        x["refseq.v"],
    ),
    axis=1,
)
print("Processing CUG")
df_uORF["CUG_uORFs"] = df_uORF.apply(
    lambda x: get_uORFs(
        x.sequence,
        x.cds_start,
        x.strand,
        x.l_exon_coordinates,
        x.chrom,
        x.gene_name,
        x["refseq.v"],
        "CTG",
    ),
    axis=1,
)
print("Processing GUG")
df_uORF["GUG_uORFs"] = df_uORF.apply(
    lambda x: get_uORFs(
        x.sequence,
        x.cds_start,
        x.strand,
        x.l_exon_coordinates,
        x.chrom,
        x.gene_name,
        x["refseq.v"],
        "GTG",
    ),
    axis=1,
)
print("Processing UUG")
df_uORF["UUG_uORFs"] = df_uORF.apply(
    lambda x: get_uORFs(
        x.sequence,
        x.cds_start,
        x.strand,
        x.l_exon_coordinates,
        x.chrom,
```

```
            x.gene_name,
            x["refseq.v"],
            "TTG",
        ),
        axis=1,
    )
```

3.5 Calculating number of uORFs and percentages for transcripts and genes which contain uORFs

In [ ]:

```
print("Number of transcripts containing AUG_uORFs:", df_uORF["AUG_uORFs"].notna().sum())
print(
    "Percent transcripts containing AUG_uORFS:",
    round(
        df_uORF["AUG_uORFs"].notna().sum()
        / (df_uORF["AUG_uORFs"].notna().sum() + df_uORF["AUG_uORFs"].isna().sum())
        * 100,
        1,
    ),
)
print(
    "Number of genes containing AUG_uORFs:",
    len(set(df_uORF.loc[df_uORF["AUG_uORFs"].notna()]["gene_name"])),
)
print(
    "Percent genes containing AUG_uORFs:",
    round(
        len(set(df_uORF.loc[df_uORF["AUG_uORFs"].notna()]["gene_name"]))
        / (
            len(set(df_uORF.loc[df_uORF["AUG_uORFs"].notna()]["gene_name"]))
            + len(set(df_uORF.loc[df_uORF["AUG_uORFs"].isna()]["gene_name"]))
        )
        * 100,
        1,
    ),
    "\n",
)

print("Number of transcripts containing CUG_uORFs:", df_uORF["CUG_uORFs"].notna().sum())
print(
    "Percent transcripts containing CUG_uORFS:",
    round(
        df_uORF["CUG_uORFs"].notna().sum()
        / (df_uORF["CUG_uORFs"].notna().sum() + df_uORF["CUG_uORFs"].isna().sum())
        * 100,
        1,
    ),
)
print(
    "Number of genes containing CUG_uORFs:",
    len(set(df_uORF.loc[df_uORF["CUG_uORFs"].notna()]["gene_name"])),
)
print(
    "Percent genes containing CUG_uORFs:",
    round(
        len(set(df_uORF.loc[df_uORF["CUG_uORFs"].notna()]["gene_name"]))
        / (
            len(set(df_uORF.loc[df_uORF["CUG_uORFs"].notna()]["gene_name"]))
            + len(set(df_uORF.loc[df_uORF["CUG_uORFs"].isna()]["gene_name"]))
        )
        * 100,
        1,
    ),
    "\n",
```

```
)

print("Number of transcripts containing GUG_uORFs:", df_uORF["GUG_uORFs"].notna().sum())
print(
    "Percent transcripts containing GUG_uORFS:",
    round(
        df_uORF["GUG_uORFs"].notna().sum()
        / (df_uORF["GUG_uORFs"].notna().sum() + df_uORF["GUG_uORFs"].isna().sum())
        * 100,
        1,
    ),
)
print(
    "Number of genes containing GUG_uORFs:",
    len(set(df_uORF.loc[df_uORF["GUG_uORFs"].notna()]["gene_name"])),
)
print(
    "Percent genes containing GUG_uORFs:",
    round(
        len(set(df_uORF.loc[df_uORF["GUG_uORFs"].notna()]["gene_name"]))
        / (
            len(set(df_uORF.loc[df_uORF["GUG_uORFs"].notna()]["gene_name"]))
            + len(set(df_uORF.loc[df_uORF["GUG_uORFs"].isna()]["gene_name"]))
        )
        * 100,
        1,
    ),
    "\n",
)

print("Number of transcripts containing UUG_uORFs:", df_uORF["UUG_uORFs"].notna().sum())
print(
    "Percent transcripts containing UUG_uORFS:",
    round(
        df_uORF["UUG_uORFs"].notna().sum()
        / (df_uORF["UUG_uORFs"].notna().sum() + df_uORF["UUG_uORFs"].isna().sum())
        * 100,
        1,
    ),
)
print(
    "Number of genes containing UUG_uORFs:",
    len(set(df_uORF.loc[df_uORF["UUG_uORFs"].notna()]["gene_name"])),
)
print(
    "Percent genes containing UUG_uORFs:",
    round(
        len(set(df_uORF.loc[df_uORF["UUG_uORFs"].notna()]["gene_name"]))
        / (
            len(set(df_uORF.loc[df_uORF["UUG_uORFs"].notna()]["gene_name"]))
            + len(set(df_uORF.loc[df_uORF["UUG_uORFs"].isna()]["gene_name"]))
        )
        * 100,
        1,
    ),
    "\n",
)
```

3.6 Creation of has_0_AUG_isoform

In [ ]:

```
df_uORF["has_0_AUG_isoform"] = df_uORF.groupby(["gene_name"])["n_AUG_uORFs"].transform(
    lambda x: (0 in list(x))
)
```

3.7 Creation of Number of 0 AUG isoforms

In [ ]:
```
df_uORF["n_0_AUG_isoform"] = df_uORF.groupby("gene_name")["n_AUG_uORFs"].transform(
    lambda x: (list(x).count(0))
)
```

In [ ]:
```
#### 3.8 at_least_1_uORF
```

In [ ]:
```
df_uORF["at_least_1_uORF"] = df_uORF.groupby("gene_name")["n_AUG_uORFs"].transform(
    lambda x: sum(list(x)) > 0
)
```

4. TWIPS Resources generation and output

Definitions:

s_n_mane_uorf_aug - where mane_select is true AUG_uORFs is not empty.

4.1. SET: s_mane_uorf_aug All the genes where the MANE variant contains a uORF with an uAUG initiation codons

In [ ]:
```
s_n_mane_uorf_aug = set(
    df_uORF.loc[
        (df_uORF["mane_select"] == True)
        & (df_uORF["AUG_uORFs"].notnull())
        & (df_uORF["n_AUG_uORFs"] > 0)
    ]["gene_name"]
)
```

In [ ]:
```
len(s_n_mane_uorf_aug)
```

In [ ]:
```
with open("../references/refseq/sets/s_n_mane_uorf_aug.pickle", "wb") as handle:
    pickle.dump(s_n_mane_uorf_aug, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

4.2. SET: s_any_uorf_aug At least 1 uORF in at least 1 transcript variant

In [ ]:
```
s_n_any_uorf_aug = set(df_uORF.loc[df_uORF["at_least_1_uORF"] == True]["gene_name"])
len(s_n_any_uorf_aug)
```

In [ ]:
```
with open("../references/refseq/sets/s_n_any_uorf_aug.pickle", "wb") as handle:
    pickle.dump(s_n_any_uorf_aug, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

4.3. SET: s_all_uorf_aug every variant must contain a uORF

In [ ]:
```
s_n_all_uorf_aug = set(df_uORF.loc[df_uORF["n_0_AUG_isoform"] == 0]["gene_name"])
```

In [ ]:
```
len(s_n_all_uorf_aug)
```

In [ ]:
```
with open("../references/refseq/sets/s_n_all_uorf_aug.pickle", "wb") as handle:
    pickle.dump(s_n_all_uorf_aug, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

In [ ]:
```
s_n_all_uorf_aug
```

5. Ribosome Profiling Analysis

5.1 Create the l_exon_coordinates column

This column contains a list of genomic coordinates 0-indexed. These coordinates can be used to query the summarized ribosome profiling and mRNA data used for the visualizaiton application.

In [ ]:
```
df_uORF["l_exon_coordinates"] = df_uORF.apply(
    lambda x: get_exon_coordinate_list(x.t_exons, x.strand), axis=1
)
```

In [ ]:
```
# verification
df_uORF.loc[df_uORF["refseq.v"] == "NM_005523.6"]["l_exon_coordinates"].item()[0]
```

In [ ]:
```
df_uORF.loc[df_uORF["refseq.v"] == "NM_152486.4"]["CUG_uORFs"].item()
```

In [ ]:
```
df_uORF.loc[df_uORF["refseq.v"] == "NM_005523.6"]["AUG_uORFs"].item()
```

In [ ]:
```
df_uORF["n_isoforms"] = df_uORF.groupby(["gene_name"])["refseq.v"].transform("count")
```

In [ ]:
```
df_uORF["test_column"] = df_uORF.apply(
    lambda x: x.l_exon_coordinates[x.cds_start], axis=1
)
```

In [ ]:
```
bw.close()
```

bigwig_file_processor

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: None

Output: Used by uORF Detective and uORF processing scripts

This script merges large bigwig files to bedgraphs and converts them to bigwig files.

In [1]:
```
import os
```

In [28]:
```
def create_list(profile_type):
    """
    Provided a bigwig type, it creates a list of all datasets for use with the
    bigWigMerge command line tool.
    The bigWigMerge command line tool outputs a bedgraph file.
    Bedgraph files can be converted to bigwig files with the bedgraphtobigwig
    tool also from UCSC.
    """
    if profile_type == "initiation":
        os.chdir(
            "/Users/nsvrzikapa/Library/CloudStorage/OneDrive-Nexus365\
            /Oxford/projects/upregulate/analysis/bioinformatics/twips/gwips_initiation"
        )
        path = "../../twips/gwips_initiation/"
        filename = "initiation.list"
    if profile_type == "footprints":
        os.chdir(
            "/Users/nsvrzikapa/Library/CloudStorage/OneDrive-Nexus365\
            /Oxford/projects/upregulate/analysis/bioinformatics/twips/gwips_footprints"
        )
        path = "../../twips/gwips_footprints/"
        filename = "footprints.list"
    if profile_type == "mRNA":
        os.chdir(
            "/Users/nsvrzikapa/Library/CloudStorage/OneDrive-Nexus365\
            /Oxford/projects/upregulate/analysis/bioinformatics/twips/gwips_mrna"
        )
        path = "../../twips/gwips_mrna/"
        filename = "mrna.list"
    if profile_type == "40s":
        os.chdir(
            "/Users/nsvrzikapa/Library/CloudStorage/OneDrive-Nexus365\
            /Oxford/projects/upregulate/analysis/bioinformatics/twips/gwips_40s"
        )
        path = "../../twips/gwips_40s/"
        filename = "40s.list"
    if profile_type == "custom":
        os.chdir(
            "/Users/nsvrzikapa/Library/CloudStorage/OneDrive-Nexus365\
            /Oxford/projects/upregulate/analysis/bioinformatics/twips/twips_mrna"
        )
        path = "../../twips/twips_mrna/"
        filename = "custom.list"

    l_bigwigs = os.listdir()
    try:
        l_bigwigs = list(filter((filename).__ne__, l_bigwigs))
    except:
        pass
```

```
    with open(filename, "w") as f:
        for item in l_bigwigs:
            f.write("%s\n" % item)
    return filename
```

1.2 Merge the lists of bigwig files

Outputs one merged file per type of bigwig data in bedGraph format. Utilizes the UCSC bigWigMerge command line.

3 files are create:

1. ns_global_initiation.bedGraph
2. ns_global_footprints.bedGraph
3. ns_global_mrna.bedGraph

In [17]:
```
list_name = create_list("initiation")
!bigWigMerge -inList  $list_name "ns_global_initiation.bedGraph"
```

```
initiation.list
Got 24 chromosomes from 8 bigWigs
Processing.......................
```

1.3 Sorting of bedGraph files

Prior to conversion to BigWig bedGraph files need to be sorted.

In [18]:
```
!sort -k1,1 -k2,2n ns_global_initiation.bedGraph > ns_global_initiation_sorted.bedGraph
```

1.4 Fetching of chromosome sizes from UCSC

In [19]:
```
!fetchChromSizes hg38 > hg38.chrom.sizes
```

```
INFO: trying CURL  for database hg38
url: http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes
```

1.5 Conversion of bedGraph files to BigWig

Once the chromosome sizes are fetched, the sorted bedGraph can be converted to a bigwig file.

In [20]:
```
!bedGraphToBigWig ns_global_initiation_sorted.bedGraph \
hg38.chrom.sizes twips_initiation.bw
```

1.6 The same process is repeated for the footprints and mrna bigwig files

footprints

In [ ]:
```
list_name = create_list("footprints")
!bigWigMerge -inList  $list_name "ns_global_footprints.bedGraph"
```

In [ ]:
```
!sort -k1,1 -k2,2n ns_global_footprints.bedGraph > ns_global_footprints_sorted.bedGraph
```

In [ ]:
```
!fetchChromSizes hg38 > hg38.chrom.sizes
```

In [ ]:
```
!bedGraphToBigWig ns_global_footprints_sorted.bedGraph hg38.chrom.sizes \
twips_footprints.bw
```

mrna

In [ ]:
```
list_name = create_list("mrna")
!bigWigMerge -inList  $list_name "ns_global_mrna.bedGraph"
```

In [ ]:
```
!sort -k1,1 -k2,2n ns_global_mrna.bedGraph > ns_global_mrna_sorted.bedGraph
```

In [ ]:
```
!fetchChromSizes hg38 > hg38.chrom.sizes
```

In [ ]:
```
!bedGraphToBigWig ns_global_mrna_sorted.bedGraph hg38.chrom.sizes twips_mrna.bw
```

40s

In [24]:
```
list_name = create_list("40s")
!bigWigMerge -inList  $list_name "ns_global_40s.bedGraph"
```

```
Got 24 chromosomes from 2 bigWigs
Processing.......................
```

In [25]:
```
!sort -k1,1 -k2,2n ns_global_40s.bedGraph > ns_global_40s_sorted.bedGraph
```

In [26]:
```
!fetchChromSizes hg38 > hg38.chrom.sizes
```

```
INFO: trying CURL  for database hg38
url: http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes
```

In [27]:
```
!bedGraphToBigWig ns_global_40s_sorted.bedGraph hg38.chrom.sizes twips_40s.bw
```

custom mRNA

In [30]:
```
list_name = create_list("custom")
!bigWigMerge -inList  $list_name "ns_global_custom.bedGraph"
```

```
Got 24 chromosomes from 4 bigWigs
Processing.......................
```

In [31]:
```
!sort -k1,1 -k2,2n ns_global_custom.bedGraph > ns_global_custom_sorted.bedGraph
```

In [32]:
```
!fetchChromSizes hg38 > hg38.chrom.sizes
```

```
INFO: trying CURL  for database hg38
url: http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes
```

In [33]:
```
!bedGraphToBigWig ns_global_custom_sorted.bedGraph hg38.chrom.sizes twips_custom.bw
```

uORF Detective 1

uORF Detective Data Preparation

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: Connection to UCSC

This notebook takes in a list of gene identifiers and collects annotations for each transcript variant associated with those identifiers from UCSC. It then processes this data and exports a text file that is used by the uORF Detective script which iteratively analyzes and generates interactive output for each variant.

In [ ]:
```
import numpy as np
import pandas as pd
```

1. Grabbing the entire UCSC refGene

In [ ]:
```
%load_ext sql
```

In [ ]:
```
%sql mysql+pymysql://genome@genome-mysql.soe.ucsc.edu/hg38?charset=utf8&use_unicode=0
```

In [ ]:
```
result = %sql SELECT name2 , name, chrom, strand, txStart, txEnd, cdsStart, \
cdsEnd, exonCount,  CONVERT(exonStarts USING utf8) AS exonStarts,\
CONVERT(exonEnds USING utf8) AS exonEnds, CONVERT(exonFrames USING utf8)\
AS exonFrames FROM ncbiRefSeqCurated;

df_refseq = result.DataFrame()
```

In [ ]:
```
df_refseq
```

2. Clean up binary strings

In [ ]:
```
b_df = df_refseq.select_dtypes([np.object])
b_df = b_df.stack().str.decode("utf-8").unstack()
for col in b_df:
    df_refseq[col] = b_df[col]
df_refseq
```

3. Filter must start with NM_

In [ ]:
```
df_nm_refseq = df_refseq[df_refseq["name"].str[0:2] == "NM"]
```

4. Enter genes identifiers of interest (ie. TP53)

In [ ]:
```
df_refseq_list = df_nm_refseq[df_nm_refseq["name2"].isin(["TP53"])]
```

5. Generate transcript variant annotation output

In [ ]:
```
df_refseq_list.sort_values(by=["name2"], inplace=True)
```

In [ ]:
```
df_refseq_list.to_csv("../twips/merge/df_nm_refseq_list.csv")
df_refseq_list.to_csv("../twips/muscle_blind/df_nm_refseq_list.csv")
```

In [ ]:
```
df_refseq_list
```

uORF Detective 2

uORF Detective Main Visualization Application

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: uORF Detective 1 output

Generates interactive visualizations of uORF in context of ribosome profiling data, visualized on a transcript level. This notebook takes in the .csv output of uORF Detective 1 It uses precalculated aggregated ribosome profiling datasets as dependencies. These files are generated with the bigwig_file_processor script. Together, this information is used to annoate uORFS, and mRNA feature coordinates. Uses the feature-viewer library to generate HTML/JS interactive output.

1. Import Libraries

In [ ]:
```
import glob
import json
import math
import os
import re
import sys

# For RenderJSON
import uuid


import numpy as np
import pandas as pd
import path
import pyBigWig

# Ensemble request
import requests
import sqlalchemy
import tabulate

# HTML header edit
from bs4 import BeautifulSoup
from IPython.core.display import HTML
from IPython.display import display, display_html, display_javascript

# Widgets
from ipywidgets import interact, widgets
```

In [ ]:
```
df_analysis = pd.DataFrame(
    columns=[
        "name2",
        "name",
        "chrom",
        "strand",
        "txStart",
        "txEnd",
        "cdsStart",
        "cdsEnd",
        "exonCount",
        "exonStarts",
        "exonEnds",
        "exonFrames",
```

```
            "t_txStart",
            "t_txEnd",
            "t_cdsStart",
            "t_cdsEnd",
            "t_exonStarts",
            "t_exonEnds",
            "t_exons",
            "max_init",
            "porf_init",
            "num_uorfs",
        ]
    )
```

2. Helper Functions

In [ ]:

```
def reverse_complement(seq):
    """
    Returns reverse complement of a given nucleic acid sequence
    """
    d = {"A": "T", "T": "A", "C": "G", "G": "C"}
    return "".join([d[x] for x in seq[::-1]])


def get_reading_frame(coordinate_uORF, coordinate_CDS):
    """
    Given a coordinates of CDS and uORF,
    Returns the reading frame of uORF relative to uORF
    Options are 0,1,2 where 1 is -1 and 2 is -2, while 0 is the same reading
    frame as the CDS.
    """
    return (coordinate_CDS - coordinate_uORF) % 3
```

3. Load Gene Table

Loads output from uORF Detective 1

In [ ]:

```
df = pd.read_csv("df_nm_refseq_list.csv", delimiter=",")

df = df.loc[:, ~df.columns.str.contains("^Unnamed")]
df
```

4. Conversions

The genomic starts and ends are inverted in genes in the - strands.

1. Create a two new columns: t_cdsStart and t_cdsEnd, t_tx_Start and t_txEnd to get the cds and transcript true start coordinates.
2. Create t_exonStarts and t_exonEnds to reverse the exon start and stop coordinates.
3. Create a list of tuples of the exon starts and ends t_exons made from the t_exonStarts and t_exonEnds

In [ ]:

```
# 1.Create a two new columns: t_cdsStart and t_cdsEnd, t_tx_Start and
# t_txEnd to get the cds and transcript true start coordinates.
df["t_txStart"] = df.apply(
    lambda x: x["txStart"] if x["strand"] == "+" else x["txEnd"], axis=1
)
df["t_txEnd"] = df.apply(
    lambda x: x["txEnd"] if x["strand"] == "+" else x["txStart"], axis=1
)
```

```
df["t_cdsStart"] = df.apply(
    lambda x: x["cdsStart"] if x["strand"] == "+" else x["cdsEnd"], axis=1
)
df["t_cdsEnd"] = df.apply(
    lambda x: x["cdsEnd"] if x["strand"] == "+" else x["cdsStart"], axis=1
)
df
```

In [ ]:
```
# 2.Create t_exonStarts and t_exonEnds to reverse the
#   exon start and stop coordinates.

# requires conversion of str to list of exon starts and ends
df["exonStarts"] = df["exonStarts"].apply(
    lambda x: [int(x) for x in x.split(",")[0:-1]]
)
df["exonEnds"] = df["exonEnds"].apply(lambda x: [int(x) for x in x.split(",")[0:-1]])
df["t_exonStarts"] = df.apply(
    lambda x: x["exonStarts"]
    if x["strand"] == "+"
    else sorted(x["exonStarts"], reverse=True),
    axis=1,
)
df["t_exonEnds"] = df.apply(
    lambda x: x["exonEnds"]
    if x["strand"] == "+"
    else sorted(x["exonEnds"], reverse=True),
    axis=1,
)
df
```

In [ ]:
```
# 3. Create a list of tuples of the exon starts and ends
#    t_exons made from the t_exonStarts and t_exonEnds
def f(x):
    temp = []
    for i, j in zip(x["t_exonStarts"], x["t_exonEnds"]):
        temp.append([i, j])
    return temp


df["t_exons"] = df.apply(f, axis=1)
# a = list(df.loc[df['name'] == "NM_000514" ]['t_exons'].values[0])
# a

df
```

In [ ]:
```
df = df.sort_values("name2", ascending=True).reset_index(drop=True)
df
```

5. Main Algorithm

Note: twoBitToFa is a command line tool and can be installed via conda, smaller to larger numerical value must be provided.

Manual overide of error throwing refseqs

PAX2/NM_003988

GNAS/NM_001077490

GNAS/NM_001309883

PHF6/NM_001015877

In [ ]:
```
l_refseqs = df["name"].tolist()
```

In [ ]:
```
# l_refseqs.remove('NM_003988')
# l_refseqs.remove('NM_001077490')
# l_refseqs.remove('NM_001309883')
# l_refseqs.remove('NM_001015877')
```

In [ ]:
```
home_dir = "/Users/nsvrzikapa/Documents/OneDrive - Nexus365\
/Oxford/projects/upregulate/analysis/bioinformatics/twips/muscle_blind/"
os.chdir(home_dir)
os.getcwd()
```

In [ ]:
```
l_num_uorfs = []
l_max_init = []
l_porf_init = []
l_problematic_refseqs = []

for i, refseq in enumerate(l_refseqs):
    print(i, refseq)
    try:
        num_uorfs = 0
        max_init = 0

        # Write Gene name and RefSeq to JS
        refseq_js = refseq + ".js"
        # Get Gene name
        gene = str(df.loc[df["name"] == refseq]["name2"].values[0])

        with open("template.html", "r") as f:
            contents = f.read()
            soup = BeautifulSoup(contents, "lxml")
        js_tag = soup.new_tag("script")
        js_tag.attrs["type"] = "text/javascript"
        # assemble string here
        js_tag.attrs["src"] = refseq_js
        soup.head.append(js_tag)
        html_out = gene + "_" + refseq + ".html"

        for a in soup.findAll("&amp;"):
            a.replaceWith("&")

        if not os.path.exists(gene + "/" + refseq):
            os.makedirs(gene + "/" + refseq)
            os.chdir(home_dir + gene + "/" + refseq)
            print(os.getcwd())
        else:
            os.chdir(home_dir + gene + "/" + refseq)
            print(os.getcwd())

        # write html
        with open(html_out, "w", encoding="utf-8") as file:
            file.write(str(soup))
        soup = ""
        # a 0-based location of the cds_START
        cds_start = int(df.loc[df["name"] == refseq]["t_cdsStart"].values[0])
        # Get CDS end
```

```
cds_end = int(df.loc[df["name"] == refseq]["t_cdsEnd"].values[0])
# Get exons coordinate list
exon_ranges = list(df.loc[df["name"] == refseq]["t_exons"].values[0])
print(exon_ranges)
# Get strand
strand = str(df.loc[df["name"] == refseq]["strand"].values[0])
# Get chromosome
chromosome = str(df.loc[df["name"] == refseq]["chrom"].values[0])

# Create the empty js file
open(refseq_js, "w").close()

file_js = open(refseq_js, "a")  # append mode
file_js.write('var gene ="' + gene + '";\n')
file_js.write('var refseq ="' + refseq + '";\n')

# Get all numbers within an exon range and assemble
# Get the sequence corresponding to each exon
# Assemble the exon string

exon_sequences = []
exon_coordinates = []

for rng in exon_ranges:
    url = (
        "../../hg38.2bit:" + chromosome + ":" + str(rng[0]) + "-" + str(rng[1])
    )
    # print(url)
    query = !twoBitToFa $url stdout
    query = "".join(query[1:]).upper()
    # print (query)
    if strand == "-":
        # print (reverse_complement(query))
        exon_sequences.append(reverse_complement(query))
        exon_coordinates.append(np.arange(rng[0] + 1, rng[1] + 1)[::-1])

    else:
        exon_sequences.append(query)
        exon_coordinates.append(np.arange(rng[0], rng[1]))
        # print (query)

# print (exon_sequences)
# print (exon_coordinates)

# QC and creating the exon coordinate track

# QC was done on positive and negative strand genes on 2 exon starts and end
# coordinates as well as start and end sequences.

# Generating the complete sequence track of the mRNA

# This is just joining the exon_sequences sequences into a string.

var_seq = "".join(exon_sequences)

# Write text to file in append mode
# Given that javascript is not able to open files due to security measures in
# browsers and this is a locally executed script, we create a pure javascript
# file that can be loaded as a script in the html.
# This way the variables are created and populated by Python but displayed in
# the local browser without the necessity of a server.

file_js.write('var seq ="' + var_seq + '";\n')

# Generation of Exon track
file_js.write(
```

```
    "var exons =[{"
    + "x:1,y:"
    + str(len(exon_sequences[0]))
    + ',"description":"Exon 1"}'
)
init_y = len(exon_sequences[0])
for i, seq in enumerate(exon_sequences[1:], 2):
    file_js.write(
        ",{x:"
        + str(init_y + 1)
        + ",y:"
        + str(init_y + len(seq))
        + ',"description":"Exon '
        + str(i)
        + '"}'
    )
    init_y += len(seq)
file_js.write("]\n")

# Generation of region track
# Gather all the data
# NOTE TWIPS JS is 1-based

# 1-based the 5'UTR always starts at position 1 by default
utr5_start = 1
# 1-based position of the end of the UTR is the length of the sequence
utr3_end = len(var_seq)
# flattens the exon_coordinates list of exon coordinates
exon_coordinates_flat = [
    coordinate for exon in exon_coordinates for coordinate in exon
]
print(exon_coordinates_flat)
# the last base of the 5'UTR in a 1-based view is the cds_start as it is 0-based
utr5_end = exon_coordinates_flat.index(cds_start)
# 1 based position of the coding sequence
mrna_cds_start = utr5_end + 1
#
mrna_cds_end = exon_coordinates_flat.index(cds_end)
utr3_start = mrna_cds_end + 1

# Write CDS track
file_js.write(
    "var utr_cds =[{x:1,y:"
    + str(utr5_end)
    + ',"description":"5\'UTR",id:"utr5",color:"#abababe"},'
    + "{x:"
    + str(mrna_cds_start)
    + ",y:"
    + str(mrna_cds_end)
    + ',"description":"Coding Region"},'
    + "{x:"
    + str(utr3_start)
    + ",y:"
    + str(utr3_end)
    + ',"description":"3\'UTR",id:"utr3",color:"#abababe"}]\n'
)

# Generation of uORF Track

utr_ATG_coordinates = [
    m.start() for m in re.finditer("ATG", var_seq[0:mrna_cds_start])
]
d_stop_codon = {"TAG": "Amber", "TGA": "Opal", "TAA": "Ochre"}
stop_codons = ["TAG", "TGA", "TAA"]

for x, start in enumerate(utr_ATG_coordinates):
```

```
    for i in range(start, len(var_seq) - 2, 3):
        evaluate = var_seq[i : i + 3]
        if evaluate in stop_codons:
            # print (x,start+1,(i+3))
            frame = get_reading_frame(start, cds_start)
            utr_ATG_coordinates[x] = (
                x + 1,
                start + 1,
                i + 3,
                d_stop_codon[evaluate],
                frame,
            )
            break

# print (utr_ATG_coordinates)
if not utr_ATG_coordinates:
    print("No uORFs")
    file_js.write("var uorfs =[]\n")
    num_uorfs = 0
else:
    num_uorfs = len(utr_ATG_coordinates)
    file_js.write(
        "var uorfs =[{x:"
        + str(utr_ATG_coordinates[0][1])
        + ",y:"
        + str(utr_ATG_coordinates[0][2])
        + ',"description":'
        + '"uORF 1'
        + "  Frame:"
        + str(utr_ATG_coordinates[0][4])
        + "  Stop:"
        + str(utr_ATG_coordinates[0][3])
        + '"}'
    )
    for uORF in utr_ATG_coordinates[1:]:
        file_js.write(
            ",{x:"
            + str(uORF[1])
            + ",y:"
            + str(uORF[2])
            + ',"description":"uORF '
            + str(uORF[0])
            + "  Frame:"
            + str(utr_ATG_coordinates[0][4])
            + "  Stop:"
            + str(utr_ATG_coordinates[0][3])
            + '"}'
        )

    file_js.write("]\n")

# Generation of uORF CTG Track
# Strong KOZAK ONLY

utr_CTG_coordinates = [
    m.start() for m in re.finditer("[AG]..CTGG", var_seq[0:mrna_cds_start])
]

for x, start in enumerate(utr_CTG_coordinates):
    for i in range(start + 3, len(var_seq) - 2, 3):
        evaluate = var_seq[i : i + 3]
        if evaluate in stop_codons:
            # print (x,start+1,(i+3))
            frame = get_reading_frame(start, cds_start)
            utr_CTG_coordinates[x] = (
                x + 1,
```

```
                start + 1 + 3,
                i + 3,
                d_stop_codon[evaluate],
                frame,
            )
            break

if not utr_CTG_coordinates:
    # print("No CTG uORFs")
    file_js.write("var uorfs_CTG =[]\n")
    num_uorfs_ctg = 0
else:
    num_uorfs_ctg = len(utr_CTG_coordinates)
    file_js.write(
        "var uorfs_CTG =[{x:"
        + str(utr_CTG_coordinates[0][1])
        + ",y:"
        + str(utr_CTG_coordinates[0][2])
        + ',"description":'
        + '"uORF_CTG 1'
        + "  Frame:"
        + str(utr_CTG_coordinates[0][4])
        + "  Stop:"
        + str(utr_CTG_coordinates[0][3])
        + '"}'
    )
    for uORF in utr_CTG_coordinates[1:]:
        file_js.write(
            ",{x:"
            + str(uORF[1])
            + ",y:"
            + str(uORF[2])
            + ',"description":"uORF_CTG '
            + str(uORF[0])
            + "  Frame:"
            + str(utr_CTG_coordinates[0][4])
            + "  Stop:"
            + str(utr_CTG_coordinates[0][3])
            + '"}'
        )

    file_js.write("]\n")

# Generation of uORF GTG Track
# Strong KOZAK ONLY

utr_GTG_coordinates = [
    m.start() for m in re.finditer("[AG]..GTGG", var_seq[0:mrna_cds_start])
]

for x, start in enumerate(utr_GTG_coordinates):
    for i in range(start + 3, len(var_seq) - 2, 3):
        evaluate = var_seq[i : i + 3]
        if evaluate in stop_codons:
            # print (x,start+1,(i+3))
            frame = get_reading_frame(start, cds_start)
            utr_GTG_coordinates[x] = (
                x + 1,
                start + 1 + 3,
                i + 3,
                d_stop_codon[evaluate],
                frame,
            )
            break

if not utr_GTG_coordinates:
```

```
    print("No GTG uORFs")
    file_js.write("var uorfs_GTG =[]\n")
    num_uorfs_gtg = 0
else:
    num_uorfs_gtg = len(utr_GTG_coordinates)
    file_js.write(
        "var uorfs_GTG =[{x:"
        + str(utr_GTG_coordinates[0][1])
        + ",y:"
        + str(utr_GTG_coordinates[0][2])
        + ',"description":'
        + '"uORF_GTG 1'
        + "  Frame:"
        + str(utr_GTG_coordinates[0][4])
        + "  Stop:"
        + str(utr_GTG_coordinates[0][3])
        + '"}'
    )
    for uORF in utr_GTG_coordinates[1:]:
        file_js.write(
            ",{x:"
            + str(uORF[1])
            + ",y:"
            + str(uORF[2])
            + ',"description":"uORF_GTG '
            + str(uORF[0])
            + "  Frame:"
            + str(utr_GTG_coordinates[0][4])
            + "  Stop:"
            + str(utr_GTG_coordinates[0][3])
            + '"}'
        )

    file_js.write("]\n")

# List of files with one element containing the global aggregates
footprint_file = glob.glob(
    "../../../../references/bigwigs/twips_footprints/*.bw"
)
init_file = glob.glob("../../../../references/bigwigs/twips_initiation/*.bw")
coverage_file = glob.glob("../../../../references/bigwigs/twips_mrna/*.bw")
twips40s_file = glob.glob("../../../../references/bigwigs/twips_40s/*.bw")
twips_custom_file = glob.glob(
    "../../../../references/bigwigs/twips_custom/*.bw"
)

l_types = [[], [], [], [], []]
footprints = l_types[0]
initiations = l_types[1]
coverage = l_types[2]
twips_40s = l_types[3]
twips_custom = l_types[4]

# create a list of opened bigwigs
bigwigs = [
    pyBigWig.open(footprint_file[0]),
    pyBigWig.open(init_file[0]),
    pyBigWig.open(coverage_file[0]),
    pyBigWig.open(twips40s_file[0]),
    pyBigWig.open(twips_custom_file[0]),
]

for coordinate in exon_coordinates_flat:
    for l, bw in zip(l_types, bigwigs):
        if math.isnan(bw.values(chromosome, coordinate - 1, coordinate)[0]):
            l.append(0)
```

```
            else:
                l.append(int(bw.values(chromosome, coordinate - 1, coordinate)[0]))

    # close all bigwigs
    for file in bigwigs:
        file.close()

    l_js_var_names = [
        "footprints",
        "initiation",
        "coverage",
        "gwips_40s",
        "twips_mrna",
    ]
    for name, track_list in zip(l_js_var_names, l_types):
        file_js.write("var " + name + " =[")
        for i, val in enumerate(track_list, 1):
            file_js.write("{x:" + str(i) + ",y:" + str(val) + "},")
        file_js.write("]\n")

    # Initiation ratios calculations
    # Get all the uORF initiation coordinates
    uorf_init_coordinates = []
    # cds_start is the pORF coordinate
    print(utr_ATG_coordinates)
    for uorf_init in utr_ATG_coordinates:
        uorf_init_coordinates.append(exon_coordinates_flat[int(uorf_init[1]) - 1])

    print(uorf_init_coordinates)
    # get cds_start coordinate
    pORF_init_coordinate = [cds_start]
    print(pORF_init_coordinate)

    l_uorf_initiations = []
    l_porf_init = []

    bw_init = pyBigWig.open(init_file[0])

    for coordinate in uorf_init_coordinates:
        if math.isnan(bw_init.values(chromosome, coordinate - 1, coordinate)[0]):
            l_uorf_initiations.append(0)
        else:
            l_uorf_initiations.append(
                int(bw_init.values(chromosome, coordinate - 1, coordinate)[0])
            )

    for coordinate in pORF_init_coordinate:
        if math.isnan(bw_init.values(chromosome, coordinate - 1, coordinate)[0]):
            l_porf_init.append(0)
        else:
            l_porf_init.append(
                int(bw_init.values(chromosome, coordinate - 1, coordinate)[0])
            )

    # mRNA section

    l_mrna_uorf = []
    l_mrna_porf = []
    bw_mrna = pyBigWig.open(coverage_file[0])

    for coordinate in uorf_init_coordinates:
        if math.isnan(bw_mrna.values(chromosome, coordinate - 1, coordinate)[0]):
            l_mrna_uorf.append(0)
        else:
            l_mrna_uorf.append(
                int(bw_mrna.values(chromosome, coordinate - 1, coordinate)[0])
```

```
        )

for coordinate in pORF_init_coordinate:
    if math.isnan(bw_mrna.values(chromosome, coordinate - 1, coordinate)[0]):
        l_mrna_porf.append(0)
    else:
        l_mrna_porf.append(
            int(bw_mrna.values(chromosome, coordinate - 1, coordinate)[0])
        )

bw_mrna.close()

print("uORF initiations", l_uorf_initiations)
print("pORF_init", l_porf_init)
print("mrna", l_mrna_uorf)
print("pORF_mrna", l_mrna_porf)
try:
    if l_porf_init[0] == 0:
        l_porf_init[0] = 1
except:
    l_porf_init[0] = None

try:
    if l_mrna_porf[0] == 0:
        l_mrna_porf[0] = 1
except:
    print("Could not get pORF mRNA coverage.")
    l_mrna_porf[0] = None

ratios = []
for i, j in zip(l_uorf_initiations, l_mrna_uorf):
    print(i, j, l_porf_init[0], l_mrna_porf[0])
    ratios.append(
        round((((i / j) / (l_porf_init[0] / l_mrna_porf[0])) * 100), 1)
    )
try:
    max_init = max(ratios)
except:
    max_init = None

file_js.write("var initiation_ratios =[")

{"UORF": 1, "name": "uORF 1", "%PORF Initiation": 15},
for i, j in enumerate(ratios, 1):
    try:
        file_js.write(
            '{"UORF":' + str(i) + ',"name":"uORF ' + str(i) + '"'
            ',"%PORF Initiation":' + str(round(j, 1)) + "},"
        )
    except:
        file_js.write(
            '{"UORF":' + str(i) + ',"name":"uORF ' + str(i) + '"'
            ',"%PORF Initiation":' + str(0) + "},"
        )
file_js.write("]\n")
file_js.close()

# Get back to home dir
os.chdir(home_dir)
try:
    print(num_uorfs, max_init, l_porf_init[0])
except:
    print("No uorfs")

l_num_uorfs.append(num_uorfs)
l_max_init.append(max_init)
```

```
        df1 = df.loc[df["name"] == refseq].copy()
        df1["max_init"] = max_init
        df1["porf_init"] = l_porf_init[0]
        df1["num_uorfs"] = num_uorfs
        # print(df1)
        df_analysis = df_analysis.append(df1)
        df_analysis.to_csv("df.analysis.csv")
        # print (df_analysis)

    except:
        os.chdir(home_dir)
        num_uorfs = None
        max_init = None
        l_porf_init[0] = None
        print(num_uorfs, max_init, l_porf_init[0])
        l_num_uorfs.append(num_uorfs)
        l_max_init.append(max_init)
        l_porf_init.append(l_porf_init[0])
        l_problematic_refseqs.append(refseq)
        print("Problems with: ", len(l_problematic_refseqs))
```

In [ ]:

```
df_analysis
```

uORF Proteome Analysis (CDFs)

Applied to Wang et al.

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: UORfeome database, proteomics data.

Analysis of uORF properties in context of human proteomics datasets. Performs statistical analysis and generates CDF plots. Outputs .csv files visualized in Graphpad Prism.

Required Packages

In [ ]:
```
import matplotlib as mpl
import pandas as pd

mpl.rcParams["figure.dpi"] = 300  # figure quality
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.api as sm

pd.set_option(
    "display.max_columns", None
)  # visualize all columns in notebook (prevents truncation)
import csv
import pickle
import time

import numpy as np
import scipy.stats as stats
from scipy.stats import kruskal
from scipy.stats import ks_2samp as ks
```

0. Helper Functions

HF1 process_tissue_data

Filter out <0.5 FPKM optional: top 10% mRNA

In [ ]:
```
def process_tissue_data(tissue, df, percent=10):
    """
    Filter out FPKM < 0.5
    Trim top 10% highest expressing transcripts (optional removed)
    Returns a set of the gene names
    """
    df = df[df[tissue] > 0.5]
    df = df.sort_values(tissue)
    df = df.reset_index(drop=True)

    # Enable these two lines to remove top 10% of highest expressing values

    to_index = len(df) - (int(len(df) / percent))
    df = df[0:to_index]
    return set(df["Gene name"])
```

1. Load the sets of Genes for filtering

In [ ]:

```
import pickle

pickle_in = open("s_genes_all_variants_uORF.pickle", "rb")
s_genes_all_variants = pickle.load(pickle_in)
pickle_in = open("s_genes_no_uORF.pickle", "rb")
s_genes_no_variants = pickle.load(pickle_in)
s_genes_hg38 = pickle.load(open("../gene_binning/s_genes_hg38.pickle", "rb"))
s_genes_mixed_variants_uORF = pickle.load(
    open("../gene_binning/s_genes_mixed_variants_uORF.pickle", "rb")
)
```

2. Load the mRNA FPKM data from Wang et al.

In [ ]:
```
df_mrna = pd.read_csv("wang_mrna.csv")
```

2.1 Check for duplicates in Gene name and remove them keep first

Note: There are duplicate values in the Gene name column.

In [ ]:
```
df_mrna.drop_duplicates(subset="Gene name", keep="first", inplace=True)
```

2.2 Filter Gene names not in hg38 db

In [ ]:
```
df_mrna = df_mrna.loc[df_mrna["Gene name"].isin(s_genes_hg38)]
```

2.3 [Option/Extra] Subset only the ubiquitously expressed mRNA

Note: May be used later should we decide to explore this subset.

In [ ]:
```
# Genes that are ubiquitously expressed
df_mrna_ubiq = df_mrna[df_mrna["Classification"] == "Expressed in all"]
```

2.4 Create a list of all the tissues from the df columns

In [ ]:
```
# A list of all represented tissues in Wang et al. 29 total.
l_tissues = list(df_mrna.columns[2:-1])
print(l_tissues)
print("Number of tissues to analyze:", len(l_tissues))
```

2.5 Create a tissue to dataframe dictionary of the raw data

Filter any value that's lower than 0.5 Reset the index.

Note: dict_tissue_to_df_mrna_processed dict holds mRNA > 0.5 FPKM as they are filtered at creation.

In [ ]:
```
# Given tissue name string returns the dataframe of that tissue
dict_tissue_to_df_mrna_processed = {}

for tissue in l_tissues:
    t_df = df_mrna[["Gene name", tissue]]
    # t_df = t_df[t_df[tissue] > 0.5]
    t_df = t_df.sort_values(by=tissue)
    t_df = t_df.reset_index(drop=True)
    dict_tissue_to_df_mrna_processed[tissue] = t_df
```

```
pickle.dump(
    dict_tissue_to_df_mrna_processed,
    open("dict_tissue_to_df_mrna_processed_unfiltered.p", "wb"),
)
pickle.dump(l_tissues, open("l_tissues_wang_unfiltered.p", "wb"))
```

2.6 Generate dict_tissue_to_set_mrna_processed

Given tissue name returns a set of genes that are expressed with > 0.5 FPKM and no duplicates

Trimming is possible if option A is enabled which uses process_tissue_data to remove the X percent of data from the top.

process_tissue_data is set to remove any FPKM < 0.5 per df and filter out the highest expressing 10% of any df

This is deactivated and as such dict_tissue_toset_mrna_processed is the set of "Gene name" from dict_tissue_to_df_mrna_processed as <0.5FPKM has already been trimmed. Activate these lines only to test with and without trimming of the highest expressing 10% of data. Otherwise do not use this function.

It can be adjusted to also remove N% highest expressing genes, by overriding the default value.

In the code to follow execute A if you want to remove top 10% expressed mRNAs B if you want to include all expressed mRNAs.

In [ ]:
```
# A
# dict_tissue_to_set_mrna_processed = {}
# for key in dict_tissue_to_df_mrna_processed.keys():
#     dict_tissue_to_set_mrna_processed[key] = process_tissue_data\
# (key,dict_tissue_to_df_mrna_processed[key],percent = 10)

# B
# Comment out and comment in above to remove N% of highest expressing genes
dict_tissue_to_set_mrna_processed = {}
for key in dict_tissue_to_df_mrna_processed.keys():
    dict_tissue_to_set_mrna_processed[key] = set(
        dict_tissue_to_df_mrna_processed[key]["Gene name"]
    )
```

3. Load the protein data

In [ ]:
```
df_protein = pd.read_csv("wang_protein.csv")
```

3.1 Exclude all genes not present in hg38 db

13308/13581 are in hg38

In [ ]:
```
df_protein = df_protein.loc[df_protein["Gene name"].isin(s_genes_hg38)]
df_protein["Gene name"].describe()
```

3.2 Drop duplicate Gene name data

13308 protein datapoints across all tissues

In [ ]:
```
df_protein.drop_duplicates(subset="Gene name", keep="first", inplace=True)
```

```
df_protein["Gene name"].describe()
```

3.3 Create a dictionary tissue to df of protein data

- For each df remove any row where protein expression is 0
- We remove the 0 values as a precaution and treat them as (no measurement).

In [ ]:
```
dict_tissue_to_df_protein_raw = {}  # all values in df must be > 0

for tissue in l_tissues:
    t_df = df_protein[["Gene name", tissue]]
    t_df = t_df[t_df[tissue] > 0]
    t_df = t_df.reset_index(drop=True)
    dict_tissue_to_df_protein_raw[tissue] = t_df
    # dict_tissue_to_set_mrna_processed[tissue] is created
    # by filtering out all FPKM < 0.5
    dict_tissue_to_df_protein_raw[tissue][
        "has_mRNA_data"
    ] = dict_tissue_to_df_protein_raw[tissue]["Gene name"].map(
        lambda x: x in dict_tissue_to_set_mrna_processed[tissue]
    )
    dict_tissue_to_df_protein_raw[tissue][
        "has_no_uORF_variants"
    ] = dict_tissue_to_df_protein_raw[tissue]["Gene name"].map(
        lambda x: x in s_genes_no_variants
    )
    dict_tissue_to_df_protein_raw[tissue][
        "all_variants_uORF"
    ] = dict_tissue_to_df_protein_raw[tissue]["Gene name"].map(
        lambda x: x in s_genes_all_variants
    )
    dict_tissue_to_df_protein_raw[tissue][
        "mixed_variants"
    ] = dict_tissue_to_df_protein_raw[tissue]["Gene name"].map(
        lambda x: x in s_genes_mixed_variants_uORF
    )
```

In [ ]:
```
print("QC: Protein data is accounted for")
print(
    "Has mRNA Data:",
    len(
        set(
            dict_tissue_to_df_protein_raw["Liver"].loc[
                dict_tissue_to_df_protein_raw["Liver"]["has_mRNA_data"] == True
            ]["Gene name"]
        )
    ),
)
print(
    "Has no uORFS:",
    len(
        set(
            dict_tissue_to_df_protein_raw["Liver"].loc[
                (dict_tissue_to_df_protein_raw["Liver"]["has_no_uORF_variants"] == True)
                & (dict_tissue_to_df_protein_raw["Liver"]["has_mRNA_data"] == True)
            ]["Gene name"]
        )
    ),
)
print(
    "All variants uORFs",
```

```
    len(
        set(
            dict_tissue_to_df_protein_raw["Liver"].loc[
                (dict_tissue_to_df_protein_raw["Liver"]["all_variants_uORF"] == True)
                & (dict_tissue_to_df_protein_raw["Liver"]["has_mRNA_data"] == True)
            ]["Gene name"]
        )
    ),
)
print(
    "Mixed uORFS/No uORFs:",
    len(
        set(
            dict_tissue_to_df_protein_raw["Liver"].loc[
                (dict_tissue_to_df_protein_raw["Liver"]["mixed_variants"] == True)
                & (dict_tissue_to_df_protein_raw["Liver"]["has_mRNA_data"] == True)
            ]["Gene name"]
        )
    ),
)
```

3.4 Intersect each protein set with filtered mRNA set

Key point:

dict_tissue_to_df_protein has every Protein matched with an mRNA datapoint with FPKM > 0.5

dict_tissue_to_df_raw is not filtered by mRNA data and contains FPKM < 0.5.

In [ ]:
```
dict_tissue_to_df_protein = {}

for tissue in l_tissues:
    dict_tissue_to_df_protein[tissue] = dict_tissue_to_df_protein_raw[tissue][
        dict_tissue_to_df_protein_raw[tissue]["has_mRNA_data"] == True
    ]
    dict_tissue_to_df_protein[tissue] = dict_tissue_to_df_protein[tissue].sort_values(
        tissue, ascending=True
    )
    dict_tissue_to_df_protein[tissue] = dict_tissue_to_df_protein[tissue].reset_index(
        drop=True
    )
```

3.5 Functions

Given a tissue name it uses the dict_tissue_to_df_protein to return a set of genes

In [ ]:
```
def generate_tissue_no_uORFs_set(tissue):
    """
    Given a tissue it returns the Gene names
    of genes where none of the variants have
    uORFs
    """
    df = dict_tissue_to_df_protein[tissue]
    res = set(df.loc[df["has_no_uORF_variants"] == True]["Gene name"])
    return res


def generate_tissue_all_uORFs_set(tissue):
    """
    Given a tissue it returns the Gene names
    of genes where All of the variants have
    uORFs
```

```
    """
    df = dict_tissue_to_df_protein[tissue]
    res = set(df.loc[df["all_variants_uORF"] == True]["Gene name"])
    return res


def generate_tissue_mixed_uORfs_set(tissue):
    """
    Given a tissue it returns the Gene names
    of genes where some of the variants have
    uORFs and some don't. (mixed)
    """
    df = dict_tissue_to_df_protein[tissue]
    res = set(df.loc[df["mixed_variants"] == True]["Gene name"])
    return res
```

3.6 QC

The df_qc_df_protein_with_mrna_data dataframe asserts for each tissue a subset of proteins that have mRNA expression data with FPKM > 0.5 and shows that each of the ('Gene name') in the protein data are a subset of the mRNA data. The %Protein column shows what percentage of the protein data was found in the mRNA data.

In [ ]:

```
lol_qc_df_protein_with_mrna_data = []
for tissue in l_tissues:
    lol_qc_df_protein_with_mrna_data.append(
        [
            tissue,
            len(dict_tissue_to_set_mrna_processed[tissue]),
            len(dict_tissue_to_df_protein_raw[tissue]["has_mRNA_data"]),
            len(set(dict_tissue_to_df_protein[tissue]["Gene name"])),
            sum(dict_tissue_to_df_protein_raw[tissue]["has_no_uORF_variants"]),
            sum(dict_tissue_to_df_protein_raw[tissue]["all_variants_uORF"]),
            sum(dict_tissue_to_df_protein_raw[tissue]["mixed_variants"]),
            len(generate_tissue_no_uORfs_set(tissue)),
            generate_tissue_no_uORfs_set(tissue),
            dict_tissue_to_df_protein_raw[tissue].loc[
                dict_tissue_to_df_protein_raw[tissue]["Gene name"].isin(
                    generate_tissue_no_uORfs_set(tissue)
                )
            ],
            len(generate_tissue_all_uORfs_set(tissue)),
            generate_tissue_all_uORfs_set(tissue),
            dict_tissue_to_df_protein_raw[tissue].loc[
                dict_tissue_to_df_protein_raw[tissue]["Gene name"].isin(
                    generate_tissue_all_uORfs_set(tissue)
                )
            ],
            len(generate_tissue_mixed_uORfs_set(tissue)),
            generate_tissue_mixed_uORfs_set(tissue),
            dict_tissue_to_df_protein_raw[tissue].loc[
                dict_tissue_to_df_protein_raw[tissue]["Gene name"].isin(
                    generate_tissue_mixed_uORfs_set(tissue)
                )
            ],
        ]
    )

    assert set(dict_tissue_to_df_protein[tissue]["Gene name"]).issubset(
        dict_tissue_to_set_mrna_processed[tissue]
    )

df_qc_df_protein_with_mrna_data = pd.DataFrame(
```

```
    lol_qc_df_protein_with_mrna_data,
    columns=[
        "Tissue",
        "mRNA",
        "Proteins",
        "mRNA_matched_Proteins",
        "none_uORF",
        "all_uORF",
        "mixed_variants",
        "mRNA+none_uORF",
        "genes_mRNA+none_uORF",
        "data_none",
        "mRNA+all_uORF",
        "genes_mRNA+all_uORF",
        "data_all",
        "mRNA+mixed_uORF",
        "genes_mRNA+mixed_uORF",
        "data_mixed",
    ],
)

# QC the sum of all, none and mixed variants lead to the
# total number of Proteins in a tissue.  Therefore all are accounted for.
# df_qc_df_protein_with_mrna_data['sum_variants'] = \
# df_qc_df_protein_with_mrna_data['none_uORF'] + \
# df_qc_df_protein_with_mrna_data['all_uORF'] + \
# df_qc_df_protein_with_mrna_data['mixed_variants']

df_qc_df_protein_with_mrna_data
```

In [ ]:

```
df_qc_df_protein_with_mrna_data.loc[
    df_qc_df_protein_with_mrna_data["Tissue"] == "Adrenal gland"
]["data_mixed"].values[0]
```

3.7 Create a dictionary to tuples of (list of protein expression values where all splice variants have a uORF, where none of the splice variants have a uORF)

This can be used, in order to run analysis for each tissue separately and get none vs all cdfs.

Note that this takes some time as there are many tissues so it is bypassed here and commented off.

In [ ]:

```
dict_tissue_to_tuples_final = {}
for tissue in l_tissues:
    # index 0 of tuples is all variants have uORF
    # index 1 of tuples is none of the variants have uORF
    dict_tissue_to_tuples_final[tissue] = (
        sorted(
            df_qc_df_protein_with_mrna_data.loc[
                df_qc_df_protein_with_mrna_data["Tissue"] == tissue
            ]["data_all"]
            .values[0][tissue]
            .tolist()
        ),
        sorted(
            df_qc_df_protein_with_mrna_data.loc[
                df_qc_df_protein_with_mrna_data["Tissue"] == tissue
            ]["data_none"]
            .values[0][tissue]
            .tolist()
        ),
    )
```

3.8 Generate a set set_mrna_processed

Contains genes that across tissues have FPKM > 0.5

There are 6724 Genes that are expressed across ALL TISSUES and have FPKM > 0.5

Discuss how to use this or if to use this with Tom.

In [ ]:
```
lol_set_mrna_processed = []

for tissue in l_tissues:
    temp_set = dict_tissue_to_set_mrna_processed[tissue]
    lol_set_mrna_processed.append(temp_set)

set_mrna_processed = set.intersection(*lol_set_mrna_processed)

print(len(set_mrna_processed))

print(set_mrna_processed)
```

4. Analysis

4.1 No uORFs vs. uORFS for every tissue

The following code generates CDF plots for every single tissue. Runs a bit slow, so commented out.

We generate Kolmogorov Smirnov p values for each one as a table. Figure generation is commented out

In [ ]:
```
p_values = []

for tissue in l_tissues:
    print(tissue)
    # no uORFs
    x1 = dict_tissue_to_tuples_final[tissue][1]
    # uORFs
    x2 = dict_tissue_to_tuples_final[tissue][0]
    ks_p = "{:.3g}".format(ks(x1, x2, alternative="two-sided", mode="asymp")[1])
    ks_d = "{:.3g}".format(ks(x1, x2, alternative="two-sided", mode="asymp")[0])
    p_values.append([tissue, ks_p, ks_d])
    print(ks_p)
with open("p_values_no_uorf_uorf_per_tissue.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerows(p_values)
```

In [ ]:
```
# for tissue in l_tissues:

#     x1 = dict_tissue_to_tuples_final[tissue][1]
#     x2 = dict_tissue_to_tuples_final[tissue][0]

#     fig, ax = plt.subplots(figsize=(8, 4))
#     #sample = np.random.uniform(0, 1, 1000)
#     ecdf1 = sm.distributions.ECDF(x1)
#     ecdf2 = sm.distributions.ECDF(x2)


#     #x_axis = np.linspace(0, max(set(x1).union(set(x2))),100)
#     x_axis = np.linspace(min(min(x1),min(x2)), max(max(x1),max(x2)),10000000)
#     #x_axis = np.linspace(min(x2), max(x2),1000000)
```

```
#     #y_axis = np.linspace(0,1,0.2)


#     y1 = ecdf1(x_axis)
#     y2 = ecdf2(x_axis)

#     print (ks(x1, x2))
#     print (stats.mannwhitneyu(x1, x2))
#     print (kruskal(x1,x2))

#     ax.grid(True)
#     #ax.legend(loc='lower left')
#     ax.set_title('Cumulative distribution of protein expression Tissue:'+ tissue)
#     ax.set_xlabel('Protein Expression')
#     ax.set_ylabel('Fraction of Genes')

#     plt.step(x_axis, y1)
#     plt.step(x_axis, y2)
#     plt.xscale("log")

#     #Change legend to monospaced


#     legend = plt.legend(['No uORF N=['+ str(len(x2)) + "]",\
#     'uORF    N=['+ str(len(x1)) + "]"], loc='lower right')
#     plt.setp(legend.texts, family='Monospace')

#     plt.savefig(tissue + '_ecdf.png')

#     plt.show()
#     plt.clf()
```

4.2 Assemble protein data across tissues: Data Aggregation

There are two types of assembly we explored.

1. Taking the median value of the proteins that are ubiquitously expressed in all 29 tissues [5400] Expressed in all.
2. Taking the median of all proteins even if they are not expressed in some tissue. [13308] Group Enriched.

For both we replace 0 with NaN and ignore Nan for the median calculation.

In [ ]:
```
df_protein.replace(0, np.nan, inplace=True)
```

In [ ]:
```
df_protein_ubiquitous = df_protein[
    df_protein["Classification"] == "Expressed in all"
].copy()
df_protein_ubiquitous["median"] = df_protein_ubiquitous.median(axis=1, skipna=True)
df_protein_ubiquitous = df_protein_ubiquitous[["Gene name", "median"]]
df_protein_ubiquitous = df_protein_ubiquitous[df_protein_ubiquitous["median"] > 0]
df_protein_ubiquitous = df_protein_ubiquitous.sort_values("median")
df_protein_ubiquitous = df_protein_ubiquitous.reset_index(drop=True)
```

In [ ]:
```
df_protein_all = df_protein.copy()
df_protein_all["median"] = df_protein_all.median(axis=1, skipna=True)
df_protein_all = df_protein_all[["Gene name", "median"]]
df_protein_all = df_protein_all[df_protein_all["median"] > 0]
df_protein_all = df_protein_all.sort_values("median")
```

```
df_protein_all = df_protein_all.reset_index(drop=True)
df_protein_all
```

In [ ]:

```
def cdf_2(
    x1,
    x1_name,
    x2,
    x2_name,
    title,
    stat="mw",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
):
    x1 = x1
    x2 = x2

    fig, ax = plt.subplots(figsize=(8, 4))
    ecdf1 = sm.distributions.ECDF(x1)
    ecdf2 = sm.distributions.ECDF(x2)
    # determine X_axis range
    # smooth determines smoothing,bin number
    x_axis = np.linspace(min(min(x1), min(x2)), max(max(x1), max(x2)), smooth)

    y1 = ecdf1(x_axis)
    y2 = ecdf2(x_axis)

    # Statistics branch
    if stat == "mw":
        print(stats.mannwhitneyu(x1, x2))
    elif stat == "ks":
        print(ks(x1, x2, alternative="two-sided", mode="asymp"))
    else:
        print(kruskal(x1, x2))

    ax.grid(True)
    # ax.legend(loc='lower left')
    ax.set_title(title)
    ax.set_xlabel("Protein Expression")
    ax.set_ylabel("Fraction of Genes")

    plt.step(x_axis, y1)
    plt.step(x_axis, y2)
    plt.xscale(xscale)

    # Change legend to monospaced
    legend = plt.legend(
        [x1_name + " N=[" + str(len(x1)) + "]", x2_name + " N=[" + str(len(x2)) + "]"],
        loc=legend_loc,
    )
    plt.setp(legend.texts, family="Monospace")

    plt.savefig("ecdf_" + "title" + ".png")

    plt.show()
    plt.clf()

    lol = [x1, x2]
    # data output for Graphpad
    with open(title + ".csv", "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerows(lol)
```

In [ ]:
```
def cdf_3(
    x1,
    x1_name,
    x2,
    x2_name,
    x3,
    x3_name,
    title,
    stat="mw",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
):
    x1 = x1
    x2 = x2
    x3 = x3

    fig, ax = plt.subplots(figsize=(8, 4))
    ecdf1 = sm.distributions.ECDF(x1)
    ecdf2 = sm.distributions.ECDF(x2)
    ecdf3 = sm.distributions.ECDF(x3)

    # determine X_axis range
    # smooth determines smoothing,bin number
    x_axis = np.linspace(
        min(min(x1), min(x2), min(x3)), max(max(x1), max(x2), max(x3)), smooth
    )

    y1 = ecdf1(x_axis)
    y2 = ecdf2(x_axis)
    y3 = ecdf3(x_axis)

    # Statistics branch
    if stat == "mw":
        print(x1_name + " vs." + x2_name, stats.mannwhitneyu(x1, x2))
        print(x1_name + " vs." + x3_name, stats.mannwhitneyu(x1, x3))
        print(x2_name + " vs." + x3_name, stats.mannwhitneyu(x2, x3))

    elif stat == "ks":
        print(
            x1_name + " vs." + x2_name,
            ks(x1, x2, alternative="two-sided", mode="asymp"),
        )
        print(
            x1_name + " vs." + x3_name,
            ks(x1, x3, alternative="two-sided", mode="asymp"),
        )
        print(
            x2_name + " vs." + x3_name,
            ks(x2, x3, alternative="two-sided", mode="asymp"),
        )
    else:
        print(x1_name + " vs." + x2_name, kruskal(x1, x2))
        print(x1_name + " vs." + x3_name, kruskal(x1, x3))
        print(x2_name + " vs." + x3_name, kruskal(x2, x3))

    ax.grid(True)
    # ax.legend(loc='lower left')
    ax.set_title(title)
    ax.set_xlabel("Protein Expression")
    ax.set_ylabel("Fraction of Genes")

    plt.step(x_axis, y1)
    plt.step(x_axis, y2)
    plt.step(x_axis, y3)
    plt.xscale(xscale)
```

```
    # Change legend to monospaced
    legend = plt.legend(
        [
            x1_name + " N=[" + str(len(x1)) + "]",
            x2_name + " N=[" + str(len(x2)) + "]",
            x3_name + " N=[" + str(len(x3)) + "]",
        ],
        loc=legend_loc,
    )
    plt.setp(legend.texts, family="Monospace")

    plt.savefig("ecdf_" + "title" + ".png")

    plt.show()
    plt.clf()

    lol = [x1, x2, x3]
    # data output for Graphpad
    with open(title + ".csv", "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerows(lol)
```

In [ ]:

```
def cdf_4(
    x1,
    x1_name,
    x2,
    x2_name,
    x3,
    x3_name,
    x4,
    x4_name,
    title,
    stat="mw",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
):
    x1 = x1
    x2 = x2
    x3 = x3
    x4 = x4

    fig, ax = plt.subplots(figsize=(8, 4))
    ecdf1 = sm.distributions.ECDF(x1)
    ecdf2 = sm.distributions.ECDF(x2)
    ecdf3 = sm.distributions.ECDF(x3)
    ecdf4 = sm.distributions.ECDF(x4)
    # determine X_axis range
    # smooth determines smoothing,bin number
    x_axis = np.linspace(
        min(min(x1), min(x2), min(x3), min(x4)),
        max(max(x1), max(x2), max(x3), max(x4)),
        smooth,
    )

    y1 = ecdf1(x_axis)
    y2 = ecdf2(x_axis)
    y3 = ecdf3(x_axis)
    y4 = ecdf4(x_axis)

    # Statistics branch
    if stat == "mw":
        print(x1_name + " vs." + x2_name, stats.mannwhitneyu(x1, x2))
```

```
    print(x1_name + " vs." + x3_name, stats.mannwhitneyu(x1, x3))
    print(x1_name + " vs." + x4_name, stats.mannwhitneyu(x1, x4))
    print(x2_name + " vs." + x3_name, stats.mannwhitneyu(x2, x3))
    print(x2_name + " vs." + x4_name, stats.mannwhitneyu(x2, x4))
    print(x3_name + " vs." + x4_name, stats.mannwhitneyu(x3, x4))
elif stat == "ks":
    print(
        x1_name + " vs." + x2_name,
        ks(x1, x2, alternative="two-sided", mode="asymp"),
    )
    print(
        x1_name + " vs." + x3_name,
        ks(x1, x3, alternative="two-sided", mode="asymp"),
    )
    print(
        x1_name + " vs." + x4_name,
        ks(x1, x4, alternative="two-sided", mode="asymp"),
    )
    print(
        x2_name + " vs." + x3_name,
        ks(x2, x3, alternative="two-sided", mode="asymp"),
    )
    print(
        x2_name + " vs." + x4_name,
        ks(x2, x4, alternative="two-sided", mode="asymp"),
    )
    print(
        x3_name + " vs." + x4_name,
        ks(x3, x4, alternative="two-sided", mode="asymp"),
    )
else:
    print(x1_name + " vs." + x2_name, kruskal(x1, x2))
    print(x1_name + " vs." + x3_name, kruskal(x1, x3))
    print(x1_name + " vs." + x4_name, kruskal(x1, x4))
    print(x2_name + " vs." + x3_name, kruskal(x2, x3))
    print(x2_name + " vs." + x4_name, kruskal(x2, x4))
    print(x3_name + " vs." + x4_name, kruskal(x3, x4))

ax.grid(True)
# ax.legend(loc='lower left')
ax.set_title(title)
ax.set_xlabel("Protein Expression")
ax.set_ylabel("Fraction of Genes")

plt.step(x_axis, y1)
plt.step(x_axis, y2)
plt.step(x_axis, y3)
plt.step(x_axis, y4)
plt.xscale(xscale)

# Change legend to monospaced
legend = plt.legend(
    [
        x1_name + " N=[" + str(len(x1)) + "]",
        x2_name + " N=[" + str(len(x2)) + "]",
        x3_name + " N=[" + str(len(x3)) + "]",
        x4_name + " N=[" + str(len(x4)) + "]",
    ],
    loc=legend_loc,
)
plt.setp(legend.texts, family="Monospace")

plt.savefig("ecdf_" + "title" + ".png")

plt.show()
plt.clf()
```

```
    lol = [x1, x2, x3, x4]
    # data output for Graphpad
    with open(title + ".csv", "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerows(lol)
```

In [ ]:

```
def cdf_5(
    x1,
    x1_name,
    x2,
    x2_name,
    x3,
    x3_name,
    x4,
    x4_name,
    x5,
    x5_name,
    title,
    stat="mw",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
):
    x1 = x1
    x2 = x2
    x3 = x3
    x4 = x4
    x5 = x5

    fig, ax = plt.subplots(figsize=(8, 4))
    ecdf1 = sm.distributions.ECDF(x1)
    ecdf2 = sm.distributions.ECDF(x2)
    ecdf3 = sm.distributions.ECDF(x3)
    ecdf4 = sm.distributions.ECDF(x4)
    ecdf5 = sm.distributions.ECDF(x5)
    # determine X_axis range
    # smooth determines smoothing,bin number
    x_axis = np.linspace(
        min(min(x1), min(x2), min(x3), min(x4), min(x5)),
        max(max(x1), max(x2), max(x3), max(x4), max(x5)),
        smooth,
    )

    y1 = ecdf1(x_axis)
    y2 = ecdf2(x_axis)
    y3 = ecdf3(x_axis)
    y4 = ecdf4(x_axis)
    y5 = ecdf5(x_axis)

    # Statistics branch
    if stat == "mw":
        print(x1_name + " vs." + x2_name, stats.mannwhitneyu(x1, x2))
        print(x1_name + " vs." + x3_name, stats.mannwhitneyu(x1, x3))
        print(x1_name + " vs." + x4_name, stats.mannwhitneyu(x1, x4))
        print(x1_name + " vs." + x5_name, stats.mannwhitneyu(x1, x5))

        print(x2_name + " vs." + x3_name, stats.mannwhitneyu(x2, x3))
        print(x2_name + " vs." + x4_name, stats.mannwhitneyu(x2, x4))
        print(x2_name + " vs." + x5_name, stats.mannwhitneyu(x2, x5))

        print(x3_name + " vs." + x4_name, stats.mannwhitneyu(x3, x4))
        print(x3_name + " vs." + x5_name, stats.mannwhitneyu(x3, x5))
```

```
    print(x4_name + " vs." + x5_name, stats.mannwhitneyu(x4, x5))
elif stat == "ks":
    print(
        x1_name + " vs." + x2_name,
        ks(x1, x2, alternative="two-sided", mode="asymp"),
    )
    print(
        x1_name + " vs." + x3_name,
        ks(x1, x3, alternative="two-sided", mode="asymp"),
    )
    print(
        x1_name + " vs." + x4_name,
        ks(x1, x4, alternative="two-sided", mode="asymp"),
    )
    print(
        x1_name + " vs." + x5_name,
        ks(x1, x5, alternative="two-sided", mode="asymp"),
    )
    print(
        x2_name + " vs." + x3_name,
        ks(x2, x3, alternative="two-sided", mode="asymp"),
    )
    print(
        x2_name + " vs." + x4_name,
        ks(x2, x4, alternative="two-sided", mode="asymp"),
    )
    print(
        x2_name + " vs." + x5_name,
        ks(x2, x5, alternative="two-sided", mode="asymp"),
    )
    print(
        x3_name + " vs." + x4_name,
        ks(x3, x4, alternative="two-sided", mode="asymp"),
    )
    print(
        x3_name + " vs." + x5_name,
        ks(x3, x5, alternative="two-sided", mode="asymp"),
    )
    print(
        x4_name + " vs." + x5_name,
        ks(x4, x5, alternative="two-sided", mode="asymp"),
    )
else:
    print(x1_name + " vs." + x2_name, kruskal(x1, x2))
    print(x1_name + " vs." + x3_name, kruskal(x1, x3))
    print(x1_name + " vs." + x4_name, kruskal(x1, x4))
    print(x1_name + " vs." + x5_name, kruskal(x1, x5))
    print(x2_name + " vs." + x3_name, kruskal(x2, x3))
    print(x2_name + " vs." + x4_name, kruskal(x2, x4))
    print(x2_name + " vs." + x5_name, kruskal(x2, x5))
    print(x3_name + " vs." + x4_name, kruskal(x3, x4))
    print(x3_name + " vs." + x5_name, kruskal(x3, x5))
    print(x4_name + " vs." + x5_name, kruskal(x4, x5))

ax.grid(True)
# ax.legend(loc='lower left')
ax.set_title(title)
ax.set_xlabel("Protein Expression")
ax.set_ylabel("Fraction of Genes")

plt.step(x_axis, y1)
plt.step(x_axis, y2)
plt.step(x_axis, y3)
plt.step(x_axis, y4)
plt.step(x_axis, y5)
plt.xscale(xscale)
```

```
    # Change legend to monospaced
    legend = plt.legend(
        [
            x1_name + " N=[" + str(len(x1)) + "]",
            x2_name + " N=[" + str(len(x2)) + "]",
            x3_name + " N=[" + str(len(x3)) + "]",
            x4_name + " N=[" + str(len(x4)) + "]",
            x5_name + " N=[" + str(len(x5)) + "]",
        ],
        loc=legend_loc,
    )
    plt.setp(legend.texts, family="Monospace")

    plt.savefig("ecdf_" + "title" + ".png")

    plt.show()
    plt.clf()

    lol = [x1, x2, x3, x4, x5]
    # data output for Graphpad
    with open(title + ".csv", "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerows(lol)
```

4.3 uORF vs. No uORF

In [ ]:
```
cdf_2(
    df_protein_ubiquitous[df_protein_ubiquitous["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "ubiq_no_uORF",
    df_protein_ubiquitous[
        df_protein_ubiquitous["Gene name"].isin(s_genes_all_variants)
    ]["median"].to_list(),
    "ubiq_all_uORF",
    "Expressed_in_all_Proteins_No_uORF_vs_All_uORF",
    stat="ks",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
)
```

In [ ]:
```
cdf_2(
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "all_no_uORF",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_all_variants)][
        "median"
    ].to_list(),
    "all_all_uORF",
    "Group_enriched__No_uORF_vs_All_uORF",
    stat="ks",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.4 Get genes where all splice variants have 1 uORF and all splice variants have more than 1 uorf

- First subset the uORF db with s_genes_all_variants this will give a list of all the genes where all splice variants have at least 1 uorf
- Some genes have variants with 1 and some have variants with more than 1.
- Need to separate the genes where all splice variants have 1, and all splice variants have more than 1

In [ ]:
```
df_db_uorfs = pd.read_csv("../gene_binning/hg38.uORF.DB.txt", delimiter="\t")
df_db_uorfs
```

In [ ]:
```
# This subsets the genes that may have
df_db_uorfs = df_db_uorfs[df_db_uorfs["GeneName"].isin(s_genes_all_variants)]
df_db_uorfs
```

In [ ]:
```
df_db_uorfs.loc[df_db_uorfs["GeneName"] == "HOXA11"]
```

4.5 Remove genes that have mixed numbers of uORFs

In [ ]:
```
df_db_uorfs_1 = df_db_uorfs[df_db_uorfs["uORFs.per.transcript"] == 1]
df_db_uorfs_1 = df_db_uorfs_1[df_db_uorfs_1["GeneName"].isin(s_genes_all_variants)]
set_db_uorfs_1 = set(df_db_uorfs_1["GeneName"].to_list())


df_db_uorfs_more_than_1 = df_db_uorfs[df_db_uorfs["uORFs.per.transcript"] > 1]
df_db_uorfs_more_than_1 = df_db_uorfs_more_than_1[
    df_db_uorfs_more_than_1["GeneName"].isin(s_genes_all_variants)
]
set_db_uorfs_more_than_1 = set(df_db_uorfs_more_than_1["GeneName"].to_list())

# means it has 1 but in some variants has > 1
set_mixed_uorf_numbers = set_db_uorfs_1.intersection(set_db_uorfs_more_than_1)

# print (len(set_db_uorfs_1),len(set_db_uorfs_more_than_1),len(set_mixed_uorf_numbers))

# overide to remove the mixed genes with some variants having 1 and some having more than
set_db_uorfs_1 = set_db_uorfs_1.difference(set_mixed_uorf_numbers)
set_db_uorfs_more_than_1 = set_db_uorfs_more_than_1.difference(set_mixed_uorf_numbers)

print(len(set_db_uorfs_1), len(set_db_uorfs_more_than_1))
```

In [ ]:
```
pickle.dump(set_db_uorfs_1, open("set_db_uorfs_1.p", "wb"))
pickle.dump(df_protein_all, open("df_protein_all.p", "wb"))
```

4.6 Binning 1-3, 3-6, >6 uORFs

In [ ]:
```
df_db_uorfs_2_to_3 = df_db_uorfs[
    (df_db_uorfs["uORFs.per.transcript"] > 1) & df_db_uorfs["uORFs.per.transcript"] < 4
]
df_db_uorfs_2_to_3 = df_db_uorfs_2_to_3[
    df_db_uorfs_2_to_3["GeneName"].isin(s_genes_all_variants)
]
set_db_uorfs_2_to_3 = set(df_db_uorfs_2_to_3["GeneName"].to_list())

df_db_uorfs_4_to_5 = df_db_uorfs[
    (df_db_uorfs["uORFs.per.transcript"] > 3)
    & (df_db_uorfs["uORFs.per.transcript"] <= 5)
]
df_db_uorfs_4_to_5 = df_db_uorfs_4_to_5[
```

```
    df_db_uorfs_4_to_5["GeneName"].isin(s_genes_all_variants)
]
set_db_uorfs_4_to_5 = set(df_db_uorfs_4_to_5["GeneName"].to_list())

df_db_uorfs_more_than_5 = df_db_uorfs[df_db_uorfs["uORFs.per.transcript"] > 5]
df_db_uorfs_more_than_5 = df_db_uorfs_more_than_5[
    df_db_uorfs_more_than_5["GeneName"].isin(s_genes_all_variants)
]
set_db_uorfs_more_than_5 = set(df_db_uorfs_more_than_5["GeneName"].to_list())

df_db_uorfs_more_than_6 = df_db_uorfs[df_db_uorfs["uORFs.per.transcript"] > 6]
df_db_uorfs_more_than_6 = df_db_uorfs_more_than_6[
    df_db_uorfs_more_than_6["GeneName"].isin(s_genes_all_variants)
]
set_db_uorfs_more_than_6 = set(df_db_uorfs_more_than_6["GeneName"].to_list())
```

4.6b PLOT: Binned genes based on number of uORFs

In [ ]:

```
cdf_5(
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_1)][
        "median"
    ].to_list(),
    "1 uORF",
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_2_to_3)][
        "median"
    ].to_list(),
    "2-3 uORFs",
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_4_to_5)][
        "median"
    ].to_list(),
    "4-5 uORFs",
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_more_than_5)][
        "median"
    ].to_list(),
    ">5 uORFs",
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_more_than_6)][
        "median"
    ].to_list(),
    ">6 uORFs",
    "uORF_number_effect_bins",
    stat="ks",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
)
```

In [ ]:

```
df_protein_all = df_protein_all.loc[df_protein_all["Gene name"].isin(s_genes_hg38)]
df_protein_all["Gene name"].describe()
```

4.7 PLOT: Expression of genes that have only 1 uORF vs Genes that have more than 1 uORF in their 5'UTR

In [ ]:

```
cdf_4(
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_all_variants)][
        "median"
    ].to_list(),
    "All uORF",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
```

```
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_1)][
        "median"
    ].to_list(),
    "1 uORF",
    df_protein_all[df_protein_all["Gene name"].isin(set_db_uorfs_more_than_1)][
        "median"
    ].to_list(),
    ">1 uORF",
    "uORF_number_effect",
    stat="ks",
    smooth=10000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.8 Evaluate the Kozak impact in genes with 1 uORF

Do it based on canonical sequences. First add canonical column to the data. Then for all the values that are the canonical pull the type of Kozak context it's under.

In [ ]:
```
# from Tom's file
df_db_uorfs_strong_kozak = df_db_uorfs[df_db_uorfs["uORF.context"] == "STRONG"]
df_db_uorfs_strong_kozak = df_db_uorfs_strong_kozak[
    df_db_uorfs_strong_kozak["GeneName"].isin(set_db_uorfs_1)
]
set_df_db_uorfs_strong_kozak = set(df_db_uorfs_strong_kozak["GeneName"])
df_protein_strong_kozak_1_uorf = df_protein_all[
    df_protein_all["Gene name"].isin(set_df_db_uorfs_strong_kozak)
]

df_db_uorfs_weak_kozak = df_db_uorfs[df_db_uorfs["uORF.context"] == "WEAK"]
df_db_uorfs_weak_kozak = df_db_uorfs_weak_kozak[
    df_db_uorfs_weak_kozak["GeneName"].isin(set_db_uorfs_1)
]
set_df_db_uorfs_weak_kozak = set(df_db_uorfs_weak_kozak["GeneName"])
df_protein_weak_kozak_1_uorf = df_protein_all[
    df_protein_all["Gene name"].isin(set_df_db_uorfs_weak_kozak)
]
```

In [ ]:
```
df_db_uorfs_strong_kozak = df_db_uorfs[df_db_uorfs["uORF.context"] == "STRONG"]
df_db_uorfs_strong_kozak = df_db_uorfs_strong_kozak[
    df_db_uorfs_strong_kozak["GeneName"].isin(set_db_uorfs_1)
]
set_df_db_uorfs_strong_kozak = set(df_db_uorfs_strong_kozak["GeneName"])
df_protein_strong_kozak_1_uorf
```

In [ ]:
```
df_db_uorfs_weak_kozak = df_db_uorfs[df_db_uorfs["uORF.context"] == "WEAK"]
df_db_uorfs_weak_kozak = df_db_uorfs_weak_kozak[
    df_db_uorfs_weak_kozak["GeneName"].isin(set_db_uorfs_1)
]
set_df_db_uorfs_weak_kozak = set(df_db_uorfs_weak_kozak["GeneName"])
len(set_df_db_uorfs_weak_kozak)
df_protein_weak_kozak_1_uorf = df_protein_all[
    df_protein_all["Gene name"].isin(set_df_db_uorfs_weak_kozak)
]
df_protein_weak_kozak_1_uorf
```

In [ ]:
```
# make sets out of weak and strong kozak
s_weak = set(df_protein_weak_kozak_1_uorf["Gene name"].to_list())
```

```
s_strong = set(df_protein_strong_kozak_1_uorf["Gene name"].to_list())
```

In [ ]:
```
s_inter = s_weak.intersection(s_strong)
s_inter
```

In [ ]:
```
df_db_uorfs_inter_kozak = df_db_uorfs[df_db_uorfs["GeneName"].isin(set_db_uorfs_1)]
df_db_uorfs_inter_kozak = df_db_uorfs_inter_kozak[
    df_db_uorfs_inter_kozak["GeneName"].isin(s_inter)
]
df_db_uorfs_inter_kozak.sort_values("GeneName")
```

In [ ]:
```
df_canonical = pd.read_csv("known_canonical.txt", sep="\t")
df_canonical.dropna()
refseq_canonical_init = df_canonical["hg38.refGene.name"].to_list()
refseq_canonical = []
for ref in refseq_canonical_init:
    ref = str(ref)
    if "," in ref:
        ref_temp = ref.replace(",", "")
        refseq_canonical.append(ref_temp)
    else:
        refseq_canonical.append(ref)

refseq_canonical.remove("NM_001297778")
refseq_canonical = set(refseq_canonical)

df_db_uorfs_inter_kozak = df_db_uorfs_inter_kozak[
    df_db_uorfs_inter_kozak["RefSeq"].isin(refseq_canonical)
]
df_db_uorfs_inter_kozak
```

In [ ]:
```
df_canonical
```

In [ ]:
```
s_inter_kozak_weak = set(
    df_db_uorfs_inter_kozak[df_db_uorfs_inter_kozak["uORF.context"] == "WEAK"][
        "GeneName"
    ].to_list()
)
s_inter_kozak_strong = set(
    df_db_uorfs_inter_kozak[df_db_uorfs_inter_kozak["uORF.context"] == "STRONG"][
        "GeneName"
    ].to_list()
)
```

In [ ]:
```
print(s_inter_kozak_weak)
```

In [ ]:
```
print(s_inter_kozak_strong)
```

In [ ]:
```
set_df_db_uorfs_weak_kozak = set_df_db_uorfs_weak_kozak.union(s_inter_kozak_weak)
set_df_db_uorfs_strong_kozak = set_df_db_uorfs_strong_kozak.union(s_inter_kozak_strong)
set_df_db_uorfs_weak_kozak.difference_update(s_inter_kozak_strong)
set_df_db_uorfs_strong_kozak.difference_update(s_inter_kozak_weak)
```

In [ ]:

```
set_df_db_uorfs_weak_kozak.intersection(set_df_db_uorfs_strong_kozak)
```

In [ ]:
```
df_protein_weak_kozak_1_uorf = df_protein_all[
    df_protein_all["Gene name"].isin(set_df_db_uorfs_weak_kozak)
]
df_protein_strong_kozak_1_uorf = df_protein_all[
    df_protein_all["Gene name"].isin(set_df_db_uorfs_strong_kozak)
]
```

In [ ]:
```
set_df_db_uorfs_weak_kozak.intersection(set_df_db_uorfs_strong_kozak)
```

In [ ]:
```
len(df_protein_weak_kozak_1_uorf)
```

In [ ]:
```
len(df_protein_strong_kozak_1_uorf)
```

In [ ]:
```
set(df_protein_strong_kozak_1_uorf["Gene name"].to_list()).intersection(
    set(df_protein_weak_kozak_1_uorf["Gene name"].to_list())
)
```

In [ ]:
```
cdf_3(
    df_protein_strong_kozak_1_uorf["median"].to_list(),
    "Strong Kozak",
    df_protein_weak_kozak_1_uorf["median"].to_list(),
    "Weak Kozak",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    "Kozak_effect",
    stat="ks",
    smooth=10000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.9 Minimal uORF Effects

In [ ]:
```
df_minimal = df_db_uorfs[df_db_uorfs["uORF.AA.FASTA"] == "M*"]
df_not_minimal = df_db_uorfs[df_db_uorfs["uORF.AA.FASTA"] != "M*"]

s_minimal = set(df_minimal["GeneName"].to_list())
s_not_minimal = set(df_not_minimal["GeneName"].to_list())
```

In [ ]:
```
s_not_minimal.difference_update(s_minimal)
s_minimal.intersection(s_not_minimal)
```

In [ ]:
```
df_s_minimal = df_protein_all[df_protein_all["Gene name"].isin(s_minimal)]
df_s_not_minimal = df_protein_all[df_protein_all["Gene name"].isin(s_not_minimal)]
```

In [ ]:
```
cdf_3(
    df_s_minimal["median"].to_list(),
    "ATG*",
```

```
    df_s_not_minimal["median"].to_list(),
    "No ATG*",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    "Minimal uORFs",
    stat="ks",
    smooth=10000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.10 TIS Span Analysis of 1 uORF population

In [ ]:
```
df_db_uorfs_1 = df_db_uorfs[df_db_uorfs["GeneName"].isin(set_db_uorfs_1)]
s_tis_span = set(
    df_db_uorfs_1[df_db_uorfs_1["TIS.span"] == "YES"]["GeneName"].to_list()
)
s_no_tis_span = set(
    df_db_uorfs_1[df_db_uorfs_1["TIS.span"] == "NO"]["GeneName"].to_list()
)
s_no_tis_span.difference_update(s_tis_span)
```

In [ ]:
```
df_s_tis_span = df_protein_all[df_protein_all["Gene name"].isin(s_tis_span)]
df_s_no_tis_span = df_protein_all[df_protein_all["Gene name"].isin(s_no_tis_span)]
```

In [ ]:
```
cdf_3(
    df_s_tis_span["median"].to_list(),
    "+ TIS SPAN",
    df_s_no_tis_span["median"].to_list(),
    "- TIS SPAN",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    "TIS SPAN EFFECTS",
    stat="ks",
    smooth=10000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.11 uORF initiation to TIS distance

In [ ]:
```
df_db_uorfs_1["start_percent"] = (
    df_db_uorfs_1["uORF.start"] / df_db_uorfs_1["CDS.start"]
) * 100
```

In [ ]:
```
df_db_uorfs_1
```

In [ ]:
```
df_db_uorfs_1["canonical"] = df_db_uorfs_1["RefSeq"].apply(
    lambda x: True if x in refseq_canonical else False
)
```

In [ ]:
```
df_db_uorfs_1.sort_values("GeneName")
```

In [ ]:
```
df_db_uorfs_1.tail(100)
```

In [ ]:
```
df_db_uorfs_1 = df_db_uorfs_1[df_db_uorfs_1["canonical"] == True]
```

In [ ]:
```
s_a_1 = set(df_db_uorfs_1[df_db_uorfs_1["start_percent"] <= 20]["GeneName"].to_list())
s_a_2 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 40) & (df_db_uorfs_1["start_percent"] > 20)
    ]["GeneName"].to_list()
)
s_a_3 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 60) & (df_db_uorfs_1["start_percent"] > 40)
    ]["GeneName"].to_list()
)
s_a_4 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 80) & (df_db_uorfs_1["start_percent"] > 60)
    ]["GeneName"].to_list()
)
s_a_5 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 100) & (df_db_uorfs_1["start_percent"] > 80)
    ]["GeneName"].to_list()
)
```

In [ ]:
```
df_s_a_1 = df_protein_all[df_protein_all["Gene name"].isin(s_a_1)]
df_s_a_2 = df_protein_all[df_protein_all["Gene name"].isin(s_a_2)]
df_s_a_3 = df_protein_all[df_protein_all["Gene name"].isin(s_a_3)]
df_s_a_4 = df_protein_all[df_protein_all["Gene name"].isin(s_a_4)]
df_s_a_5 = df_protein_all[df_protein_all["Gene name"].isin(s_a_5)]
```

In [ ]:
```
s_b_1 = set(df_db_uorfs_1[df_db_uorfs_1["start_percent"] <= 10]["GeneName"].to_list())
s_b_2 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 20) & (df_db_uorfs_1["start_percent"] > 10)
    ]["GeneName"].to_list()
)
s_b_3 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 30) & (df_db_uorfs_1["start_percent"] > 20)
    ]["GeneName"].to_list()
)
s_b_4 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 40) & (df_db_uorfs_1["start_percent"] > 30)
    ]["GeneName"].to_list()
)
s_b_5 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 50) & (df_db_uorfs_1["start_percent"] > 40)
    ]["GeneName"].to_list()
)
s_b_6 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 60) & (df_db_uorfs_1["start_percent"] > 50)
    ]["GeneName"].to_list()
)
```

```
s_b_7 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 70) & (df_db_uorfs_1["start_percent"] > 60)
    ]["GeneName"].to_list()
)
s_b_8 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 80) & (df_db_uorfs_1["start_percent"] > 70)
    ]["GeneName"].to_list()
)
s_b_9 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 90) & (df_db_uorfs_1["start_percent"] > 80)
    ]["GeneName"].to_list()
)
s_b_10 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 100) & (df_db_uorfs_1["start_percent"] > 90)
    ]["GeneName"].to_list()
)

df_s_b_1 = df_protein_all[df_protein_all["Gene name"].isin(s_b_1)]
df_s_b_2 = df_protein_all[df_protein_all["Gene name"].isin(s_b_2)]
df_s_b_3 = df_protein_all[df_protein_all["Gene name"].isin(s_b_3)]
df_s_b_4 = df_protein_all[df_protein_all["Gene name"].isin(s_b_4)]
df_s_b_5 = df_protein_all[df_protein_all["Gene name"].isin(s_b_5)]
df_s_b_6 = df_protein_all[df_protein_all["Gene name"].isin(s_b_6)]
df_s_b_7 = df_protein_all[df_protein_all["Gene name"].isin(s_b_7)]
df_s_b_8 = df_protein_all[df_protein_all["Gene name"].isin(s_b_8)]
df_s_b_9 = df_protein_all[df_protein_all["Gene name"].isin(s_b_9)]
df_s_b_10 = df_protein_all[df_protein_all["Gene name"].isin(s_b_10)]
```

In [ ]:

```
cdf_5(
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    df_s_a_1["median"].to_list(),
    "0-20%",
    df_s_a_2["median"].to_list(),
    "20-40%",
    df_s_a_3["median"].to_list(),
    "40-60%",
    df_s_a_4["median"].to_list(),
    "60-80%",
    # df_s_a_5['median'].to_list(),
    # "80-100%",
    "uORF_initiation_position_20s",
    stat="ks",
    smooth=10000000,
    xscale="log",
    legend_loc="lower right",
)
```

In [ ]:

```
s_c_1 = set(df_db_uorfs_1[df_db_uorfs_1["start_percent"] <= 40]["GeneName"].to_list())
s_c_2 = set(
    df_db_uorfs_1[
        (df_db_uorfs_1["start_percent"] <= 100) & (df_db_uorfs_1["start_percent"] > 40)
    ]["GeneName"].to_list()
)
df_s_c_1 = df_protein_all[df_protein_all["Gene name"].isin(s_c_1)]
df_s_c_2 = df_protein_all[df_protein_all["Gene name"].isin(s_c_2)]
```

```
cdf_3(
    df_s_c_1["median"].to_list(),
    "0-40% INIT",
    df_s_c_2["median"].to_list(),
    "40-100% INIT",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    "Initiation Position Effects (40%)",
    stat="ks",
    smooth=10000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.12 Kozak separation of 40-100% uORFs

In [ ]:
```
# 40-100% is s_c_2
s_c_2_weak_kozak = s_c_2.intersection(set_df_db_uorfs_weak_kozak)
s_c_2_strong_kozak = s_c_2.intersection(set_df_db_uorfs_strong_kozak)

df_s_c_2_weak = df_protein_all[df_protein_all["Gene name"].isin(s_c_2_weak_kozak)]
df_s_c_2_strong = df_protein_all[df_protein_all["Gene name"].isin(s_c_2_strong_kozak)]

cdf_3(
    df_s_c_2_weak["median"].to_list(),
    "40-100% Weak Kozak",
    df_s_c_2_strong["median"].to_list(),
    "40-100% Strong Kozak",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    "Initiation Position Effects (40-100% KOZAK)",
    stat="ks",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
)
```

In [ ]:
```
# 40-100% is s_c_2
s_c_1_weak_kozak = s_c_1.intersection(set_df_db_uorfs_weak_kozak)
s_c_1_strong_kozak = s_c_1.intersection(set_df_db_uorfs_strong_kozak)

df_s_c_1_weak = df_protein_all[df_protein_all["Gene name"].isin(s_c_1_weak_kozak)]
df_s_c_1_strong = df_protein_all[df_protein_all["Gene name"].isin(s_c_1_strong_kozak)]

cdf_3(
    df_s_c_1_weak["median"].to_list(),
    "0-40% Weak Kozak",
    df_s_c_1_strong["median"].to_list(),
    "0-40% Strong Kozak",
    df_protein_all[df_protein_all["Gene name"].isin(s_genes_no_variants)][
        "median"
    ].to_list(),
    "No uORF",
    "Initiation Position Effects (0-40% KOZAK)",
    stat="ks",
    smooth=1000000,
    xscale="log",
    legend_loc="lower right",
)
```

4.13 uORF length

In [ ]:
```
df_sm_51 = df_db_uorfs[df_db_uorfs['uORF.length'] < 51]
df_la_51 = df_db_uorfs[df_db_uorfs['uORF.length'] >= 51]


s_sm_51 = set(df_sm_51['GeneName'].to_list())
s_la_51 = set(df_la_51['GeneName'].to_list())

print(len(s_sm_35))
print (len(s_la_35))

df_prot_sm_51 = df_protein_all[df_protein_all['Gene name'].isin(s_sm_51)]
df_prot_sm_51 = df_prot_sm_51[df_prot_sm_51['Gene name'].isin(set_db_uorfs_1)]
df_prot_la_51 = df_protein_all[df_protein_all['Gene name'].isin(s_la_51)]
df_prot_la_51 = df_prot_la_51[df_prot_la_51['Gene name'].isin(set_db_uorfs_1)]

s_df_prot_sm_51 = set(df_prot_sm_51['Gene name'].to_list())
s_df_prot_la_51 = set(df_prot_la_51['Gene name'].to_list())
s_length_intersect = s_df_prot_la_51.intersection(s_df_prot_sm_51)
print ("Common:",len (s_length_intersect))

s_df_prot_sm_51.difference_update(s_length_intersect)
s_df_prot_la_51.difference_update(s_length_intersect)

df_prot_sm_51 = df_protein_all[df_protein_all['Gene name'].isin(s_df_prot_sm_51)]
df_prot_la_51 = df_protein_all[df_protein_all['Gene name'].isin(s_df_prot_la_51)]

print (len(df_prot_sm_51))
print (len(df_prot_la_51))
print (len(df_prot_sm_51) + len(df_prot_la_51))
```

In [ ]:
```
cdf_3(   df_protein_all[df_protein_all['Gene name'].isin(s_genes_no_variants)]['median'].t
      "No uORF",
   df_prot_sm_51['median'].to_list(),
      "< 51",
      df_prot_la_51['median'].to_list(),
      "> 51",
      "uORF_length",

      stat = "ks",
      smooth = 10000000,xscale = "log",legend_loc = 'lower right')
```

uORF Transcriptome Analysis (Heatmaps)

Applied to Wang et al. and Jiang et al. datasets

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Dependencies: uORF Detective 1 output

Checks what percentage of genes in each percentile have uORFs by evaluating how many are a subset of either:

s_genes_all_variants (every variant has at least 1 uORF) s_genes_no_variants (none of the variants have any uORFs) s_genes_mixed_variants (some variants contain at least 1 uORF, while others have 0 uORFs)

For figure purposes the output was visualized using Graphpad Prism

In [ ]:
```
import csv
import glob
import pickle

import numpy as np
import pandas as pd
import seaborn as sns
```

1. Load the ENSG symbol to HGNC translation Dictionary

In [ ]:
```
# Load the d_refseq_fna_fasta dictionary
with open("../references/gene/dictionaries/d_ensg_to_hgnc.pickle", "rb") as handle:
    d_ensg_to_hgnc = pickle.load(handle)
```

2. Load RNA Seq data to a dataframe

Note: Log2(TPM) values were converted to TPM. The median expression value per tissue was used.

In [ ]:
```
df_jiang_mrna = pd.read_csv("../references/gene/expression/jiang_et_al_TPM.csv")
df_jiang_mrna
```

2.1 Convert the Ensemble gene IDs to HGNC symbols

Note: Assertion for uniqueness was completed anad passed with 12230 unique gene symbols.

In [ ]:
```
df_jiang_mrna["gene_name"] = df_jiang_mrna["gene.id"].map(d_ensg_to_hgnc)
```

In [ ]:
```
l_tissues_unfiltered = df_jiang_mrna.columns.to_list()
to_filter_out = ["gene.id", "gene_name"]
l_tissues_jiang = [
    tissue for tissue in l_tissues_unfiltered if tissue not in to_filter_out
]
l_tissues_jiang
```

2.2 Given tissue name string returns the dataframe of that tissue

In [ ]:
```
d_tissue_to_df_mrna_sorted_jiang = {}
```

```
for tissue in l_tissues:
    t_df = df_jiang_mrna[["gene_name", tissue]]
    t_df = t_df[t_df[tissue] > 0.5]
    t_df = t_df.sort_values(by=tissue)
    t_df = t_df.reset_index(drop=True)
    d_tissue_to_df_mrna_sorted_jiang[tissue] = t_df

pickle.dump(
    d_tissue_to_df_mrna_sorted_jiang,
    open("d_tissue_to_df_mrna_sorted_jiang.pickle", "wb"),
)
pickle.dump(l_tissues_jiang, open("l_tissues_jiang.pickle", "wb"))
```

In [ ]:

```
s_genes_all_variants_uORF = pickle.load(open("s_genes_all_variants_uORF.pickle", "rb"))
s_genes_none_variants_uORF = pickle.load(open("s_genes_no_uORF.pickle", "rb"))
s_genes_mixed_variants_uORF = pickle.load(
    open("../gene_binning/s_genes_mixed_variants_uORF.pickle", "rb")
)
s_genes_hg38 = pickle.load(open("../gene_binning/s_genes_hg38.pickle", "rb"))
```

3. Filtering of data with of any gene identifier not in hg38 db

In [ ]:

```
for tissue in l_tissues:
    d_tissue_to_df_mrna_sorted_jiang[tissue] = d_tissue_to_df_mrna_sorted_jiang[
        tissue
    ].loc[d_tissue_to_df_mrna_sorted_jiang[tissue]["gene_name"].isin(s_genes_hg38)]
```

In [ ]:

```
d_tissue_to_df_mrna_sorted_jiang["Liver"]
```

4. Main algorithm

In [ ]:

```
def test_uORF_presence(d, percentiles=100, mixed=False):
    """
    Tests uORF presence in 100 percentiles (default) can be overriden.
    Takes in a dictionary of dataframes per tissue of FPKM values.
    DFs are sorted by FPKM.
    Returns a list of Genes that have uORFs in that percentile.
    Outputs list to file: percentiles_uofs.csv for further analysis.
    """
    results_uorfs = {}
    results_no_uorfs = []

    for tissue in l_tissues:
        # Get the length of the DF
        df_length = len(d[tissue])
        # Split length in percentiles (default 100)
        # i.e. if len 200 there will be 100 bins of len 2
        # Note that the dfs are ordered by FPKM values so the first bin will be lowest exp
        split_length_to_100 = np.array_split(np.arange(df_length), percentiles)
        results_uorfs[tissue] = []

        for rng in split_length_to_100:
            # subset a mini df for each range
            df = d[tissue][rng[0] : rng[-1]]
            # gene names of dataframe to test (set)
            # This is the set of genes that needs to be evaluated if these genes contain u
            set_to_test = set(df["gene_name"])
```

```
                # This is the test
                all_variants = round(
                    len(set_to_test.intersection(s_genes_all_variants_uORF))
                    / len(set_to_test)
                    * 100,
                    1,
                )
                none_variants = round(
                    len(set_to_test.intersection(s_genes_none_variants_uORF))
                    / len(set_to_test)
                    * 100,
                    1,
                )
                mixed_variants = round(
                    len(set_to_test.intersection(s_genes_mixed_variants_uORF))
                    / len(set_to_test)
                    * 100,
                    1,
                )
                # Uncomment if you don't filter above with hg38 to see gene names that are not
                # print (all_variants + none_variants + mixed_variants)
                # complete_union = s_genes_all_variants_uORF.union(s_genes_none_variants_uORF,
                #s_genes_mixed_variants_uORF)
                # not_found = set_to_test.difference(complete_union)
                # print (not_found)

                if mixed == False:
                    # results_uorfs.append((tissue,all_variants,\
                    #set_to_test.difference(s_genes_all_variants_uORF)))
                    results_uorfs[tissue].append(all_variants)

                else:
                    # results_uorfs.append((tissue,all_variants + mixed_variants))
                    results_uorfs[tissue].append(all_variants + mixed_variants)

        return results_uorfs
```

In [ ]:
```
test_uORF_presence(d_tissue_to_df_mrna_sorted_jiang, percentiles=100, mixed=False)
```

In [ ]:
```
df_result = pd.DataFrame.from_dict(
    test_uORF_presence(d_tissue_to_df_mrna_sorted_jiang, percentiles=100, mixed=False)
)
```

In [ ]:
```
df_result
```

In [ ]:
```
df_result.to_csv("jiang_percentiles_uorfs.csv")
```

uORF Deep Learning

Author: Nenad Svrzikapa, University of Oxford, Department of Paediatrics

Note: This notebook uses python 2.7, use dl_27 environment to run it.

Dependencies: Requires training data from Sample et al for scaler, requires Sample et al. varying length model.

Varying_length_25to100_model.hdf5

This notebook uses random sequence generation sampling to probe the Sample et al. Deep Learning LSTM model to examine specific uORF properties which were subsequently experimentaly validated.

Authorship of code: This analysis uses portions of published code from Sample et al. which is required to load the model. Original analysis code starts with the counting of sections.

This code outputs sequence logo SVG images, and matplotlib images generated by the Python analysis.

Required libraries

In [23]:
```
import random

import keras
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
import seaborn as sns
from sklearn import preprocessing

np.random.seed(1337)

from collections import Counter

# logomaker import
import logomaker
from keras.layers.convolutional import Conv1D
from keras.layers.core import Activation, Dense, Dropout, Flatten
from keras.models import Sequential
from keras.optimizers import RMSprop
from keras.preprocessing import sequence

# displays logos inline within the notebook;
# remove if using a python interpreter instead
%matplotlib inline


####
# Allows for display of all columns of a df in the jupyter notebook
pd.set_option("display.max_columns", None)
```

Helper functions and dictionaries

Name	Type	Description
test_data	Function	Sample et al. This function is used to feed in a df and generate the prediction column

Name	Type	Description
one_hot_encode	Function	Sample et al. This function is used to output one_hot_encoded sequences of n padded sequences of len 100
amino_acids	List	List of all AA codons
d_aa	Dictionary	Codon to One letter Amino Acid dictionary
d_aa_properties_a	Dictionary	Single letter AA to hydrophobic, polar, positive, negative, special
d_aa_properties_b	Dictionary	Single letter AA to charged, polar, non-polar
d_aa_properties_c	Dictionary	Single letter AA to charge, polar, special, hydrophobic
l_aa	List	A list of all 20 single letter AA codes
translate_to_aa	Function	Takes string in ATGC format and a translation dictionary to return amino acid sequence
translate_to_properties	Function	Takes string in ATGC format and a translation dictionary to return amino acid sequence
logo_maker	Function	Calls the logo_maker function to make a logo, uses data from make_logo
make_logo	Function	Calculates all the Counters per position to create matrices used for logo_maker

Output files

Name	Type	Description
britt_preds_b.csv	csv	Predictions on dual-glo luciferase models
two_million_predictions_10_AA.csv	csv/df	Output of df_model the 2 million sampling predictions
df_model_ttg.csv	csv/df	Output of the df_model_ttg predictions of the df_model ATG sequences

Code Credit: The following two functions are source code from the Sample et al. training notebook and are used to encode the sequences and create the prediction column. The manuscript has been cited in the thesis.

In [24]:
```
def test_data(df, model, test_seq, obs_col, output_col="pred"):
    """
    Predict mean ribosome load using model and test set UTRs
    """

    # Scale the test set mean ribosome load
    scaler = preprocessing.StandardScaler()
    scaler.fit(df[obs_col].values.reshape(-1, 1))

    # Make predictions
    predictions = model.predict(test_seq).reshape(-1)

    # Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
    df.loc[:, output_col] = scaler.inverse_transform(predictions)
    return df


def one_hot_encode(df, col="utr100", seq_len=100):
    # Dictionary returning one-hot encoding of nucleotides.
    nuc_d = {
        "a": [1, 0, 0, 0],
        "c": [0, 1, 0, 0],
        "g": [0, 0, 1, 0],
        "t": [0, 0, 0, 1],
        "n": [0, 0, 0, 0],
```

```
    }

    # Create empty matrix.
    vectors = np.empty([len(df), seq_len, 4])

    # Iterate through UTRs and one-hot encode
    for i, seq in enumerate(df[col].str[:seq_len]):
        seq = seq.lower()
        a = np.array([nuc_d[x] for x in seq])
        vectors[i] = a
    return vectors
```

1. Load the appropriate trained model 25-100nt limit

In [25]:
```
# For the standard scaler
df1 = pd.read_csv("../data/GSM4084997_varying_length_25to100.csv")
df = df1[df1["set"] == "random"]
df = df[df["total_reads"] >= 10]
df["utr100"] = df["utr"].str.rjust(100, "N")
```

In [26]:
```
model = keras.models.load_model(
    "/Users/nsvrzikapa/Documents/OneDrive - Nexus365\
    /Oxford/projects/upregulate/analysis\
    /machine_learning/sample_2019_models\
    /modeling/saved_models/Varying_length_25to100_model.hdf5"
)
```

2. Fit the scaler

Note: the data is trained on MRL values in a certain range, thus the model has to be scaled to the center of that data. Predictions are then rescaled back to MRL values. This sets all predictions within the range of the dataset that was used to train the model.

In [27]:
```
scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))
```

Out[27]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

3. Initial predictions for dual-glo luciferase data

Note: All of the code from this point was developed by Nenad Svrzikapa.

In [28]:
```
# Removed some sequences that were longer than 100 nucleotides.
df_b = pd.read_csv(
    "/Users/nsvrzikapa/Documents/OneDrive - Nexus365\
    /Oxford/projects/upregulate/analysis/machine_learning\
    /sample_2019_models/Data/HOXA11_functional_mutant_UTR_sequences_b.csv"
)
df_b["len"] = df_b["sequences"].apply(lambda x: len(x))

# take only sequences <=100 nt
df_test = df_b[df_b["len"] <= 100].copy()
df_test.reset_index(inplace=True, drop=True)
# Create column and pad the sequences to 100 with Ns if they are shorter
df_test["utr100"] = df_test["sequences"].str.rjust(100, "N")
# one_hot_encode the sequences
e_df_test = one_hot_encode(df_test, seq_len=100)
```

```
predictions = model.predict(e_df_test).reshape(-1)
df_test["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_test.to_csv(
    "/Users/nsvrzikapa/Documents/OneDrive - Nexus365\
    /Oxford/projects/upregulate/analysis/machine_learning\
    /sample_2019_models/Data/britt_preds_b.csv"
)
```

4. Simulations

10 Amino Acid sequences randomly generated to replace the HOXA11 uORF 5 times X 2 million sequences were generated with similar findings.

HOXA11 Template sequence for 6 AA and 10AA simulations

In [30]:
```
wt_a = "NNNNNNNNNNNNCTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGAATCATG"
wt_b = "GAGAGCCCAAGGTAGCCCAATA"  # to simulate 6 AA
wt_c = "TAGCCCAATA"  # to simulate 10 AA
```

4.1 Amino Acid nomenclature and properties lists and dictionaries

In [31]:
```
# no stop codons or methionine in amino_acids
amino_acids = [
    "GCT",
    "GCC",
    "GCA",
    "GCG",  # Alanine Ala A
    "CGT",
    "CGC",
    "CGA",
    "CGG",
    "AGA",
    "AGG",  # Arginine Arg R
    "GAT",
    "GAC",  # Aspartic Acid D
    "AAT",
    "AAC",  # Asparagine ASN N
    "TGT",
    "TGC",  # Cysteine Cys C
    "GAA",
    "GAG",  # Glutamate Glu E
    "CAA",
    "CAG",  # Glutamine Gln Q
    "GGT",
    "GGC",
    "GGA",
    "GGG",  # Glycine Gly G
    "CAT",
    "CAC",  # Histidine His H
    "ATT",
    "ATC",
    "ATA",  # Isoleucine Ile I
    "CTT",
    "CTC",
    "CTA",
    "CTG",
    "TTA",
    "TTG",  # Leucine Leu L
    "AAA",
    "AAG",  # Lysine Lys K
```

```
    "TTT",
    "TTC",  # Phenylalainine Phe F
    "CCT",
    "CCC",
    "CCA",
    "CCG",  # Proline Pro P
    "TCT",
    "TCC",
    "TCA",
    "TCG",
    "AGT",
    "AGC",  # Serine Ser S
    "ACT",
    "ACC",
    "ACA",
    "ACG",  # Threonine Thr T
    "TGG",  # Tryptophan Trp W
    "TAT",
    "TAC",  # Tyrosine Tyr Y
    "GTT",
    "GTC",
    "GTA",
    "GTG",  # Valine Val V
    "ATG",
]  # Methionine

d_aa = {
    "GCT": "A",
    "GCC": "A",
    "GCA": "A",
    "GCG": "A",
    "CGT": "R",
    "CGC": "R",
    "CGA": "R",
    "CGG": "R",
    "AGA": "R",
    "AGG": "R",
    "AAT": "N",
    "AAC": "N",
    "GAT": "D",
    "GAC": "D",
    "TGT": "C",
    "TGC": "C",
    "CAA": "Q",
    "CAG": "Q",
    "GAA": "E",
    "GAG": "E",
    "GGT": "G",
    "GGC": "G",
    "GGA": "G",
    "GGG": "G",
    "CAT": "H",
    "CAC": "H",
    "ATT": "I",
    "ATC": "I",
    "ATA": "I",
    "CTT": "L",
    "CTC": "L",
    "CTA": "L",
    "CTG": "L",
    "TTA": "L",
    "TTG": "L",
    "AAA": "K",
    "AAG": "K",
    "TTT": "F",
    "TTC": "F",
```

```
    "CCT": "P",
    "CCC": "P",
    "CCA": "P",
    "CCG": "P",
    "TCT": "S",
    "TCC": "S",
    "TCA": "S",
    "TCG": "S",
    "AGT": "S",
    "AGC": "S",
    "ACT": "T",
    "ACC": "T",
    "ACA": "T",
    "ACG": "T",
    "TGG": "W",
    "TAT": "Y",
    "TAC": "Y",
    "GTT": "V",
    "GTC": "V",
    "GTA": "V",
    "GTG": "V",
    "TAG": "",
    "TAA": "",
    "TGA": "",
    "ATG": "M",
}

# hydrophobic, polar, positive, negative, special

d_aa_properties_a = {
    "K": "+",
    "R": "+",
    "H": "+",
    "D": "-",
    "E": "-",
    "S": "P",
    "T": "P",
    "N": "P",
    "Q": "P",
    "G": "S",
    "C": "S",
    "P": "S",
    "Y": "H",
    "A": "H",
    "V": "H",
    "L": "H",
    "I": "H",
    "M": "H",
    "W": "H",
    "F": "H",
}
# charged, polar, or nonpolar
d_aa_properties_b = {
    "K": "C",
    "R": "C",
    "H": "C",
    "D": "C",
    "E": "C",
    "S": "P",
    "T": "P",
    "Y": "P",
    "N": "P",
    "Q": "P",
    "G": "N",
    "A": "N",
    "V": "N",
```

```
    "C": "N",
    "P": "N",
    "L": "N",
    "I": "N",
    "M": "N",
    "W": "N",
    "F": "N",
}
# charged, polar, special, hydrophobic
d_aa_properties_c = {
    "K": "C",
    "R": "C",
    "H": "C",
    "D": "C",
    "E": "C",
    "S": "P",
    "T": "P",
    "N": "P",
    "Q": "P",
    "C": "S",
    "G": "S",
    "P": "S",
    "A": "H",
    "V": "H",
    "P": "H",
    "L": "H",
    "I": "H",
    "M": "H",
    "W": "H",
    "F": "H",
    "Y": "H",
}

l_aa = [
    "A",
    "R",
    "N",
    "D",
    "C",
    "Q",
    "E",
    "G",
    "H",
    "I",
    "L",
    "K",
    "F",
    "P",
    "S",
    "T",
    "W",
    "Y",
    "V",
    "M",
]
l_aa_properties_a = ["+", "-", "P", "S", "H"]
l_aa_properties_b = ["C", "P", "N"]
l_aa_properties_c = ["C", "P", "S", "H"]


l_E = ["GAA", "GAG"]
l_S = ["TCT", "TCC", "TCA", "TCG", "AGT", "AGC"]
l_K = ["AAA", "AAG"]
l_W = ["TGG"]
l_R = ["CGT", "CGC", "CGA", "CGG", "AGA", "AGG"]
l_D = ["GAT", "GAC"]
```

```
l_Y = ["TAT", "TAC"]
l_A = ["GCT", "GCC", "GCA", "GCG"]
l_P = ["CCT", "CCC", "CCA", "CCG"]
```

4.2 Functions

In [33]:
```
def translate_to_aa(s, d):
    """
    Takes string in ATGC format and a translation dictionary
    to return amino acid sequence
    """
    i = 0
    l = len(s) - 15
    result = ""
    for i in xrange(3, l, 3):
        result += d[s[i : i + 3]]
    return result


def translate_to_properties(s, d):
    """
    Takes string in ATGC format and a translation dictionary
    to return amino acid sequence
    """
    result = ""
    for i in xrange(0, 6):
        result += d[s[i]]
    return result
```

In [34]:
```
### For 10 AA
def translate_to_aa(s, d):
    """
    Takes string in ATGC format and a translation dictionary
    to return amino acid sequence
    """
    i = 0
    l = len(s) - 5
    result = ""
    for i in xrange(0, l, 3):
        result += d[s[i : i + 3]]
    return result


def translate_to_properties(s, d):
    """
    Takes string in ATGC format and a translation dictionary
    to return amino acid sequence
    """
    result = ""
    for i in xrange(0, 11):
        result += d[s[i]]
    return result
```

In [35]:
```
def logo_maker(df_input, name):
    # Logomaker
    # create Logo object
    svg_name = name
    logo = logomaker.Logo(
        df_input, color_scheme="skylign_protein",
        font_name="Arial Rounded MT Bold"
    )
```

```
    # style using Logo methods
    logo.style_spines(visible=False)
    logo.style_spines(spines=["left", "bottom"], visible=True)
    logo.style_xticks(rotation=0, fmt="%d", anchor=0)

    # style using Axes methods
    logo.ax.set_ylabel("Frequency", labelpad=-1, fontsize=17)
    logo.ax.xaxis.set_ticks_position("none")
    logo.ax.xaxis.set_tick_params(pad=-1)
    logo.ax.tick_params(axis="both", which="major", labelsize=15)
    logo.fig.savefig(svg_name)


def make_logo(df_complete, logo_type, slice_type, slice_number, name):
    """
    Outputs a sequence logo based on df head or tail information.
    """
    # Slices from df based on analysis of head or tail
    if slice_type == "head":
        df = df_complete.head(slice_number)
    else:  # tail
        df = df_complete.tail(slice_number)

    # A list of counter objects for each position
    counters_aa = []
    counters_pa = []
    counters_pb = []
    counters_pc = []

    for i in range(0, 11):
        counters_aa.append(Counter([x[i] for x in df["translation"].values]))
        counters_pa.append(Counter([x[i] for x in df["aa_properties_a"].values]))
        counters_pb.append(Counter([x[i] for x in df["aa_properties_b"].values]))
        counters_pc.append(Counter([x[i] for x in df["aa_properties_c"].values]))

    # Seed dataframes
    df_final_aa = pd.DataFrame(
        columns=[
            "A",
            "R",
            "N",
            "D",
            "C",
            "Q",
            "E",
            "G",
            "H",
            "I",
            "L",
            "K",
            "F",
            "P",
            "S",
            "T",
            "W",
            "Y",
            "V",
            "M",
        ]
    )
    df_final_pa = pd.DataFrame(columns=["+", "-", "P", "S", "H"])
    df_final_pb = pd.DataFrame(columns=["C", "P", "N"])
    df_final_pc = pd.DataFrame(columns=["C", "P", "S", "H"])

    for i in range(0, 11):
```

```
            temp_aa = []
            temp_pa = []
            temp_pb = []
            temp_pc = []

            for x in l_aa:
                temp_aa.append(round(counters_aa[i][x] / float(slice_number) * 100, 1))
            for x in l_aa_properties_a:
                temp_pa.append(round(counters_pa[i][x] / float(slice_number) * 100, 1))
            for x in l_aa_properties_b:
                temp_pb.append(round(counters_pb[i][x] / float(slice_number) * 100, 1))
            for x in l_aa_properties_c:
                temp_pc.append(round(counters_pc[i][x] / float(slice_number) * 100, 1))

            df_final_aa.loc[i] = temp_aa
            df_final_pa.loc[i] = temp_pa
            df_final_pb.loc[i] = temp_pb
            df_final_pc.loc[i] = temp_pc

        # 1 index instead of 0
        df_final_aa.index += 1
        df_final_pa.index += 1
        df_final_pb.index += 1
        df_final_pc.index += 1

        # return df
        if logo_type == "aa":
            logo_maker(df_final_aa, name)
            return df_final_aa
        elif logo_type == "pa":
            logo_maker(df_final_pa, name)
            return df_final_pa
        elif logo_type == "pb":
            logo_maker(df_final_pb, name)
            return df_final_pb
        else:
            logo_maker(df_final_pc, name)
            return df_final_pc
```

4.3 HOXA11 sampling all 10AA

2000000 possibilities were samples multiple times. Replacing the sequence of the uORF of the HOXA11 uORF

In [36]:
```
# Randomly draw
s = set([])
for i in range(0, 2000000):
    s.add(
        wt_a
        + "".join([amino_acids[random.randint(0, 60)] for i in range(0, 10)])
        + wt_c
    )
```

In [37]:
```
df_model = pd.DataFrame(list(s))
df_model.rename(columns={0: "utr100"}, inplace=True)
```

In [38]:
```
df_model["aa"] = df_model["utr100"].apply(lambda x: x[57:93])
```

In [39]:
```
e_df_model = one_hot_encode(df_model, seq_len=100)
```

```
# Fit the scaler on the training data ranges
scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))
predictions = model.predict(e_df_model).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
df_model["preds"] = pd.Series(scaler.inverse_transform(predictions))
# sort the values
df_model.sort_values("preds", inplace=True)
```

In [40]:
```
df_model["translation"] = df_model["aa"].apply(lambda x: translate_to_aa(x, d_aa))
df_model["aa_properties_a"] = df_model["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_a)
)
df_model["aa_properties_b"] = df_model["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_b)
)
df_model["aa_properties_c"] = df_model["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_c)
)
```

In [57]:
```
df_model.to_csv("two_million_predictions_10_AA_new_Methionine.csv")
```

In [2]:
```
df_model = pd.read_csv("two_million_predictions_10_AA_new.csv")
```

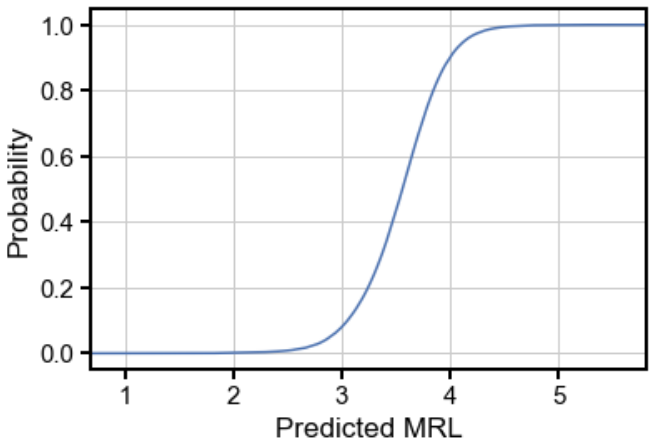
In [ ]:
```
make_logo(df_model, "pc", "head", 100, "2mil_HEAD_100_pc_methionine.svg")
```

In [ ]:
```
make_logo(df_model, "aa", "tail", 100, "2mil_TAIL_100_aa_Methionine.svg")
```

In [355…
```
df_model["cdf"] = df_model["preds"].rank(method="average", pct=True)

ax = df_model.plot(x="preds", y="cdf", grid=True, legend=False)
ax.set_ylabel("Probability", labelpad=5, fontsize=17)
ax.set_xlabel("Predicted MRL", labelpad=5, fontsize=17)

ax.tick_params(axis="both", which="major", labelsize=15)
plt.savefig("two_million_predictions.svg")
```



In [268…
```
# remove empty spaces from names
df_test.name = df_test.name.str.replace(" ", "")
```

4.4 X, Y of TTG of df_model

Replacing the 5' and the ATG of the uORF initiation with TTG containing sequence. Calling the model predictions on the same sequences with a TTG mutant.

In [545…
```
df_model_ttg = df_model[["utr100"]].copy()
df_model_ttg["utr100"] = e_df_model_ttg["utr100"].apply(
    lambda x: x.replace(
        "TCAAAGAGGCAGCTGCAGTGGAGAATCATG", "TCAAAGAGGCAGCTGCAGTGGAGAATCTTG"
    )
)
```

In [ ]:
```
e_df_model_ttg = one_hot_encode(df_model_ttg, seq_len=100)
# Fit the scaler on the training data ranges
# scaler = preprocessing.StandardScaler()
# scaler.fit(df['rl'].values.reshape(-1,1))
predictions = model.predict(e_df_model_ttg).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
df_model_ttg["preds"] = pd.Series(scaler.inverse_transform(predictions))
# sort the values
# df_model_ttg.sort_values('preds', inplace = True)
```

In [550…
```
# df_model_ttg['translation'] = df_model_ttg['aa'].apply(lambda x: translate_to_aa(x, d_a
# df_model_ttg['aa_properties_a'] = df_model_ttg['translation'].apply(lambda x: translate_
# df_model_ttg['aa_properties_b'] = df_model_ttg['translation'].apply(lambda x: translate_
# df_model_ttg['aa_properties_c'] = df_model_ttg['translation'].apply(lambda x: translate_

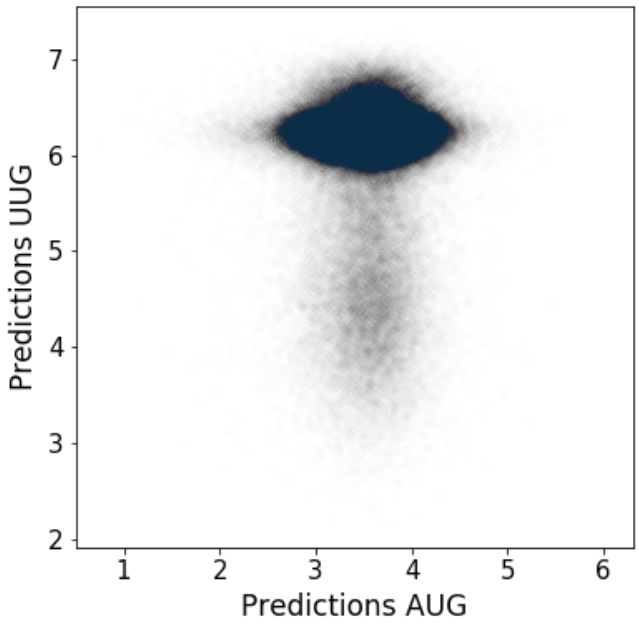
df_model_ttg.to_csv("df_model_ttg.csv")
```

In [3]:
```
df_model_ttg = pd.read_csv("df_model_ttg.csv")
```

In [4]:
```
df_model["preds_ttg"] = df_model_ttg["preds"]
```

In [168…
```
df_model_sampled = df_model.sample(200000)
ax1 = df_model_sampled.plot.scatter(
    x="preds", y="preds_ttg", c="#0C2D48", alpha=0.005, figsize=(6, 6)
)
ax1.set_ylabel("Predictions UUG", labelpad=5, fontsize=17)
ax1.set_xlabel("Predictions AUG", labelpad=5, fontsize=17)

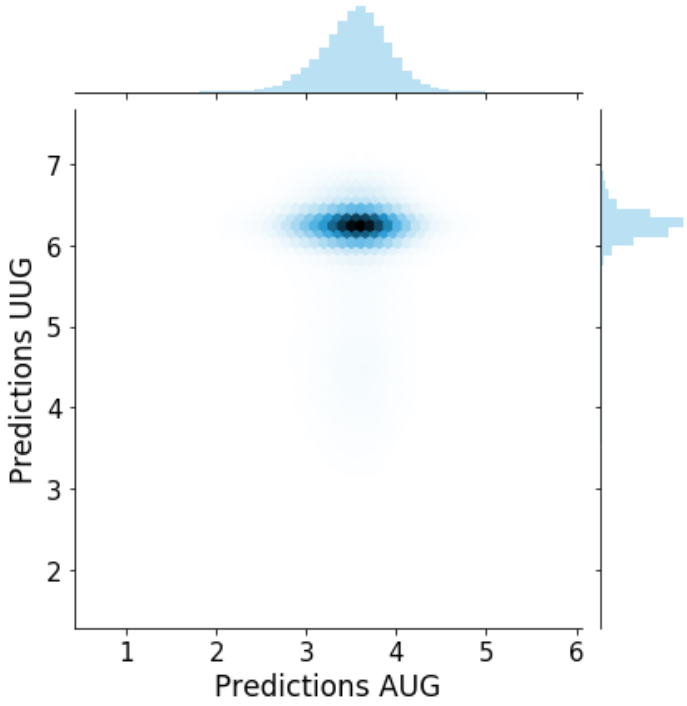
ax1.tick_params(axis="both", which="major", labelsize=15)
plt.xlim([0.5, 6.3])
plt.savefig("two_million_predictions_AUG_UUG_Sampled_200000.png", dpi=300)
```



In [61]:

```
x = df_model["preds"]
y = df_model["preds_ttg"]

ax2 = sns.jointplot(x=x, y=y, kind="hex", color="#53b3e3")
ax2.set_axis_labels("Predictions AUG", "Predictions UUG", fontsize=17)
ax2.ax_joint.tick_params(axis="both", which="major", labelsize=15)
plt.savefig("two_million_predictions_AUG_UUG_full_jointplot.svg")
```



5. Alanine scan of most inhibitory sequence found from 2 million simulations in HOXA11

In [100…

```
seq_2mil_inh = df_model.iloc[[0]]["utr100"].item()[12:]  # Ns before 12
seq_2mil_inh_TTG = seq_2mil_inh.replace(
    "ATGGACAGGCCAAGATGGAGGCGGCGCCCGGAGTAG", "TTGGACAGGCCAAGATGGAGGCGGCGCCCGGAGTAG"
)  # TTG version
seq_2mil_inh.index("ATGGACAGGCCAAGATGGAGGCGGCGCCCGGAGTAG")  # position
```

Out[100…

```
45
```

In [ ]:

```
seq_alanine_walk = []
seq_alanine_walk.extend(
    [seq_2mil_inh, seq_2mil_inh_TTG]
)  # first add the most inhibitory and ttg version of it
seq_alanine_walk

template = seq_2mil_inh
rep_list = [
    "ATGGCAAGGCCAAGATGGAGGCGGCGCCCGGAGTAG",
    "ATGGACGCACCAAGATGGAGGCGGCGCCCGGAGTAG",
    "ATGGACAGGGCAAGATGGAGGCGGCGCCCGGAGTAG",
    "ATGGACAGGCCAGCATGGAGGCGGCGCCCGGAGTAG",
    "ATGGACAGGCCAAGAGCAAGGCGGCGCCCGGAGTAG",
    "ATGGACAGGCCAAGATGGGCACGGCGCCCGGAGTAG",
    "ATGGACAGGCCAAGATGGAGGGCACGCCCGGAGTAG",
    "ATGGACAGGCCAAGATGGAGGCGGGCACCGGAGTAG",
    "ATGGACAGGCCAAGATGGAGGCGGCGCGCAGAGTAG",
    "ATGGACAGGCCAAGATGGAGGCGGCGCCCGGCATAG",
    "ATGCGATATGGAAGATGGAGGCGGCGCCCGGAGTAG",  # RYG +4C
]
for seq in rep_list:
    seq_alanine_walk.append(
        seq_2mil_inh.replace("ATGGACAGGCCAAGATGGAGGCGGCGCCCGGAGTAG", seq)
    )
seq_alanine_walk

seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCA\
    TGGACAGGCCCAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # ATG -3C
seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGAATCA\
    TGCACAGGCCCAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # ATG +4C
seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCA\
    TGCACAGGCCCAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # ATG -3C +4C

seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCT\
    TGGACAGGCCCAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # TTG -3C
seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGAATCT\
    TGCACAGGCCCAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # TTG +4C
seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCT\
    TGCACAGGCCCAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # TTG -3C +4C

seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCT\
    TGCACAGGCCCAGAGCAAGGCGGCGCCCGGAGTAGCCCAATA"
```

```
)  # TTG -3C +4C 5W->A
seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCT\
    TGCGATATGGAAGATGGAGGCGGCGCCCGGAGTAGCCCAATA"
)  # TTG -3C +4C RYG

seq_alanine_walk.append(
    "CTACTTCACGGATCCGCTTCAAAGAGGCAGCTGCAGTGGAGACTCT\
    TGCGATATGGAAGAGCAAGGCGGCGCCCGGAGTAGCCCAATA"
)  # TTG -3C +4C RYG 5W->A


seq_alanine_walk
```

In [211…
```
df_inh_alanine_walk = pd.DataFrame(seq_alanine_walk)
df_inh_alanine_walk.rename(columns={0: "utr100"}, inplace=True)
```

In [212…
```
df_inh_alanine_walk["utr100"] = df_inh_alanine_walk["utr100"].str.rjust(100, "N")
```

In [214…
```
e_df_inh_alanine_walk = one_hot_encode(df_inh_alanine_walk, seq_len=100)
# Fit the scaler on the training data ranges
# scaler = preprocessing.StandardScaler()
# scaler.fit(df['rl'].values.reshape(-1,1))

predictions = model.predict(e_df_inh_alanine_walk).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
df_inh_alanine_walk["preds"] = pd.Series(scaler.inverse_transform(predictions))
# sort the values
# df_inh_alanine_walk.sort_values('preds', inplace = True)

df_inh_alanine_walk["aa"] = df_inh_alanine_walk["utr100"].apply(lambda x: x[57:93])
df_inh_alanine_walk["translation"] = df_inh_alanine_walk["aa"].apply(
    lambda x: translate_to_aa(x, d_aa)
)
df_inh_alanine_walk["aa_properties_a"] = df_inh_alanine_walk["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_a)
)
df_inh_alanine_walk["aa_properties_b"] = df_inh_alanine_walk["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_b)
)
df_inh_alanine_walk["aa_properties_c"] = df_inh_alanine_walk["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_c)
)
```

In [216…
```
df_inh_alanine_walk.to_csv("df_inh_alanine_walk.csv")
```

In [ ]:
```
df_model_top5 = df_model.head(5)
df_model_bottom5 = df_model.tail(5)
df_inh_alanine_walk_MR_TTG = df_inh_alanine_walk.iloc[[1]]
df_inh_alanine_walk_MR_W_A = df_inh_alanine_walk.iloc[[6]]
df_inh_alanine_walk_MR_RYG = df_inh_alanine_walk.iloc[[12]]

df_export_britt = pd.concat(
    [
        df_model_top5,
        df_model_bottom5,
        df_inh_alanine_walk_MR_TTG,
        df_inh_alanine_walk_MR_W_A,
```

```
        df_inh_alanine_walk_MR_RYG,
    ]
)
df_export_britt
```

In [130…
```
df_export_britt.drop(
    ["aa_properties_a", "aa_properties_b", "aa_properties_c"], inplace=True, axis=1
)
```

In [147…
```
df_export_britt["name"] = [
    "HOXA11_ML_MR_1",
    "HOXA11_ML_MR_2",
    "HOXA11_ML_MR_3",
    "HOXA11_ML_MR_4",
    "HOXA11_ML_MR_5",
    "HOXA11_ML_LR_1",
    "HOXA11_ML_LR_2",
    "HOXA11_ML_LR_3",
    "HOXA11_ML_LR_4",
    "HOXA11_ML_LR_5",
    "HOXA11_ML_MR_1_TTG",
    "HOXA11_ML_MR_1_W_to_A",
    "HOXA11_ML_MR_1_RYG",
]
```

In [138…
```
df_export_britt.rename({"aa": "uorf_sequence"}, axis=1, inplace=True)
```

In [141…
```
df_export_britt["complete_utr_sequence"] = df_export_britt["utr100"].apply(
    lambda x: x[12:]
)
```

In [143…
```
df_export_britt["utr_length"] = df_export_britt["complete_utr_sequence"].apply(
    lambda x: len(x)
)
```

In [144…
```
df_export_britt.drop(["utr100"], inplace=True, axis=1)
```

In [149…
```
df_export_britt.to_csv("machine_learning_cloning.csv")
```

6. Create a merged df from preds for HOXA11 WT and MU, and the ML random generation

In [ ]:
```
df_britt_a = df_test.loc[df_test["name"].isin(["HOXA11_TTG", "HOXA11_WT"])]
df_2mil_head_1 = df_model.iloc[[0]]
df_2mil_head_1["sequences"] = df_2mil_head_1["utr100"].apply(lambda x: x[12:])
df_2mil_head_1["len"] = df_2mil_head_1["sequences"].apply(lambda x: len(x))

df_2mil_head_1.loc[1114706, "name"] = "HOXA11_ML_INH"
df_2mil_tail_1 = df_model.iloc[[-1]]
df_2mil_tail_1["sequences"] = df_2mil_tail_1["utr100"].apply(lambda x: x[12:])
df_2mil_tail_1.loc[126339, "name"] = "HOXA11_ML_NO_INH"
df_2mil_tail_1["len"] = df_2mil_tail_1["sequences"].apply(lambda x: len(x))
```

In [ ]:
```
df_2mil_head_1 = df_2mil_head_1[["name", "sequences", "len", "utr100", "preds"]]
df_2mil_tail_1 = df_2mil_tail_1[["name", "sequences", "len", "utr100", "preds"]]
l_df_merge = [df_britt_a, df_2mil_head_1, df_2mil_tail_1]
df_merged = pd.concat(l_df_merge)
df_merged["aa"] = df_merged["utr100"].apply(lambda x: x[54:93])
df_merged["translation"] = df_merged["aa"].apply(lambda x: translate_to_aa(x, d_aa))
df_merged.to_csv("df_merged.csv")
```

7. Modeling AA codon replacements

In [342…
```
# MDRPRWRRRPE
s_top = set([])
for i in range(0, 1000000):
    s_top.add(
        wt_a
        + l_D[random.randint(0, 1)]
        + l_R[random.randint(0, 5)]
        + l_P[random.randint(0, 3)]
        + l_R[random.randint(0, 1)]
        + l_W[0]
        + l_R[random.randint(0, 5)]
        + l_R[random.randint(0, 1)]
        + l_R[random.randint(0, 1)]
        + l_P[random.randint(0, 3)]
        + l_E[random.randint(0, 1)]
        + wt_c
    )
```

In [343…
```
s_top = set(s_top)
len(s_top)
```

Out[343… 18432

In [344…
```
df_model_top = pd.DataFrame(list(s_top))
```

In [345…
```
df_model_top.rename(columns={0: "utr100"}, inplace=True)
```

In [347…
```
df_model_top["aa"] = df_model_top["utr100"].apply(lambda x: x[54:93])
```

In [72]:
```
len("ATGCGGGCGGGTAAGATCGATGAGAGCCCAAGGTAGCCCAATA")
len("ATGACGGACCGCGATATAATTGAACTTGGGGAGTAG")
```

Out[72]: 36

In [348…
```
e_df_model_top = one_hot_encode(df_model_top, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_model_top).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
df_model_top["preds"] = pd.Series(scaler.inverse_transform(predictions))
```

In [358…
```
df_model_top["translation"] = df_model_top["aa"].apply(
    lambda x: translate_to_aa(x, d_aa)
)
df_model_top["sequences"] = df_model_top["utr100"].apply(lambda x: x[12:])
```

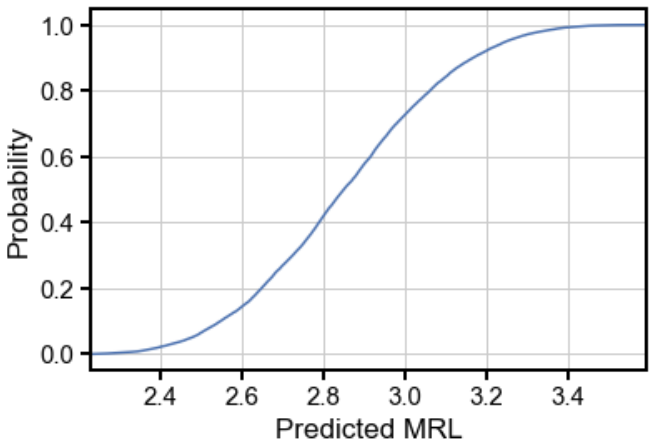
In [351…
```
df_model_top["aa_properties_a"] = df_model_top["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_a)
)
df_model_top["aa_properties_b"] = df_model_top["translation"].apply(
    lambda x: translate_to_properties(x, d_aa_properties_b)
)
```

In [356…
```
df_model_top.sort_values("preds", inplace=True)
```

In [357…
```
df_model_top["cdf"] = df_model_top["preds"].rank(method="average", pct=True)

ax_top = df_model_top.plot(x="preds", y="cdf", grid=True, legend=False)
ax_top.set_ylabel("Probability", labelpad=5, fontsize=17)
ax_top.set_xlabel("Predicted MRL", labelpad=5, fontsize=17)

ax_top.tick_params(axis="both", which="major", labelsize=15)
plt.savefig("codon_substitutions.svg")
```



In [ ]:
```
df_cs_head_1 = df_model_top.iloc[[0]]
df_cs_head_1["len"] = df_cs_head_1["sequences"].apply(lambda x: len(x))
df_cs_head_1.loc[4120, "name"] = "HOXA11_ML_CS_INH"

df_cs_tail_1 = df_model_top.iloc[[-1]]
df_cs_tail_1.loc[8791, "name"] = "HOXA11_ML_CS_LOW_INH"
df_cs_tail_1["len"] = df_cs_tail_1["sequences"].apply(lambda x: len(x))

df_cs_head_1 = df_cs_head_1[["name", "sequences", "len", "utr100", "preds"]]
df_cs_tail_1 = df_cs_tail_1[["name", "sequences", "len", "utr100", "preds"]]
l_df_merge_cs = [df_merged, df_cs_head_1, df_cs_tail_1]
df_merged_cs = pd.concat(l_df_merge_cs)
df_merged_cs["aa"] = df_merged_cs["utr100"].apply(lambda x: x[54:93])
df_merged_cs["translation"] = df_merged_cs["aa"].apply(
    lambda x: translate_to_aa(x, d_aa)
)
df_merged_cs.to_csv("df_merged_cs.csv")
```

8. Modeling of Minimal uORFs

SAMD11 5'UTR is < 100 bases long and contains no uORFs.

GCAGAGCCCAGCAGATCCCTGCGGCGTTCGCGAGGGTGGGACGGGAAGCGGGCTGGGAAGTCGGGCCGAGGGAA

In [374…

```
samd11wt_utr5 = "GCAGAGCCCAGCAGATCCCTGCGGCGTTCGCGAGGGTGGGACGGG\
AAGCGGGCTGGGAAGTCGGGCCGAGGGAAAAGTCTGAAGACGCTT"
l_modeled_samd11_minimal = []

l_modeled_samd11_minimal.append(samd11wt_utr5)
for i, base in enumerate(samd11wt_utr5):
    seq = samd11wt_utr5[0:i] + "ATGTAG" + samd11wt_utr5[i:]
    l_modeled_samd11_minimal.append(seq)
```

In [375…

```
isg15wt_utr5 = (
    "GGCGGCTGAGAGGCAGCGAACTCATCTTTGCCAGTACAGGAGCTTGTGCCGTGGCCCAC\
    AGCCCACAGCCCACAGCC"
)
l_modeled_isg15_minimal = []

l_modeled_isg15_minimal.append(isg15wt_utr5)
for i, base in enumerate(isg15wt_utr5):
    seq = isg15wt_utr5[0:i] + "ATGTAG" + isg15wt_utr5[i:]
    l_modeled_isg15_minimal.append(seq)
```

In [376…

```
acot11wt_utr5 = (
    "ATTTGGAGTCAGGCCTGGCTGTTGCTCAGGTGACCAGCTTGTGTCTCTGGGAGGGCGCTG\
    CTTTCCCCGGCCACCCGGCGCG"
)
l_modeled_acot11_minimal = []

l_modeled_acot11_minimal.append(acot11wt_utr5)
for i, base in enumerate(acot11wt_utr5):
    seq = acot11wt_utr5[0:i] + "ATGTAG" + acot11wt_utr5[i:]
    l_modeled_acot11_minimal.append(seq)
```

In [388…

```
hes5wt_utr5 = "GTCCGCGCCAGCCCGGGACGCGCTTGGCCTTGCCCGCGCCCGCTCGCCTCGT\
CTCGCCCGGCCTCCCCGCGTCGCCTCGTCGCCTGTTCCGCGCCAGGC"
l_modeled_hes5_minimal = []

l_modeled_hes5_minimal.append(hes5wt_utr5)
for i, base in enumerate(acot11wt_utr5):
    seq = hes5wt_utr5[0:i] + "ATGTAG" + hes5wt_utr5[i:]
    l_modeled_hes5_minimal.append(seq)
```

In [389…

```
vamp3wt_utr5 = "GTCCCACCCATCTCCCTGGCCTCCGGTCCCAACTTCGCTTCTCTGCTGACC\
CTCTCTCGTCGCCGCTGCCGCCGCCGCAGCTGCCAAA"
l_modeled_vamp3_minimal = []

l_modeled_vamp3_minimal.append(vamp3wt_utr5)
for i, base in enumerate(acot11wt_utr5):
    seq = vamp3wt_utr5[0:i] + "ATGTAG" + vamp3wt_utr5[i:]
    l_modeled_vamp3_minimal.append(seq)
```

In [391…

```
hmx3wt_utr5 = (
```

```
    "GATTTCCGGCCTCTCGCCTGCTCGCCCCGCCGCCCGCCTGTCCCGCTCCCTCCCTCCCGGGGA\
    CCCGGAGGAGAGGGGACC"
)

l_modeled_hmx3_minimal = []

l_modeled_hmx3_minimal.append(hmx3wt_utr5)
for i, base in enumerate(acot11wt_utr5):
    seq = hmx3wt_utr5[0:i] + "ATGTAG" + hmx3wt_utr5[i:]
    l_modeled_hmx3_minimal.append(seq)
```

In [392…

```
df_samd11 = pd.DataFrame(l_modeled_samd11_minimal, columns=["sequences"])
df_isg15 = pd.DataFrame(l_modeled_isg15_minimal, columns=["sequences"])
df_acot11 = pd.DataFrame(l_modeled_acot11_minimal, columns=["sequences"])
df_hes5 = pd.DataFrame(l_modeled_hes5_minimal, columns=["sequences"])
df_vamp3 = pd.DataFrame(l_modeled_vamp3_minimal, columns=["sequences"])
df_hmx3 = pd.DataFrame(l_modeled_hmx3_minimal, columns=["sequences"])
```

In [393…

```
df_hes5.reset_index(inplace=True, drop=True)
df_hes5["utr100"] = df_hes5["sequences"].str.rjust(100, "N")
e_df_test_hes5 = one_hot_encode(df_hes5, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_test_hes5).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
df_hes5["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_hes5.preds.to_csv("hes5_preds.csv", header=True)
```

In [397…

```
df_vamp3.reset_index(inplace=True, drop=True)
df_vamp3["utr100"] = df_vamp3["sequences"].str.rjust(100, "N")
e_df_test_vamp3 = one_hot_encode(df_vamp3, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_test_vamp3).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
df_vamp3["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_vamp3.preds.to_csv("vamp3_preds.csv", header=True)
```

In [395…

```
df_hmx3.reset_index(inplace=True, drop=True)
df_hmx3["utr100"] = df_hmx3["sequences"].str.rjust(100, "N")
e_df_test_hmx3 = one_hot_encode(df_hmx3, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_test_hmx3).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
```

```
df_hmx3["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_hmx3.preds.to_csv("hmx3_preds.csv", header=True)
```

In [396…

```
df_samd11.reset_index(inplace=True, drop=True)
df_samd11["utr100"] = df_samd11["sequences"].str.rjust(100, "N")
e_df_test_samd11 = one_hot_encode(df_samd11, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_test_samd11).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
df_samd11["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_samd11.preds.to_csv("samd11_preds.csv", header=True)
```

In [418…

```
df_isg15.reset_index(inplace=True, drop=True)
df_isg15["utr100"] = df_isg15["sequences"].str.rjust(100, "N")
e_df_test_isg15 = one_hot_encode(df_isg15, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_test_isg15).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
df_isg15["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_isg15.preds.to_csv("isg15_preds.csv", header=True)
```

In [399…

```
df_acot11.reset_index(inplace=True, drop=True)
df_acot11["utr100"] = df_acot11["sequences"].str.rjust(100, "N")
e_df_test_acot11 = one_hot_encode(df_acot11, seq_len=100)

scaler = preprocessing.StandardScaler()
scaler.fit(df["rl"].values.reshape(-1, 1))

predictions = model.predict(e_df_test_acot11).reshape(-1)

# Inverse scaled predicted mean ribosome load and return in a column labeled 'pred'
# df_test['output_col'] = predictions
# df_test
df_acot11["preds"] = pd.Series(scaler.inverse_transform(predictions))
df_acot11.preds.to_csv("acot11_preds.csv", header=True)
```

Organize for merging

SAMD11

In [ ]:

```
# samd11, acot11,isg15
df_samd11_wt = df_samd11.iloc[[0]]
df_samd11_wt["sequences"] = df_samd11_wt["utr100"].apply(lambda x: x[10:])

df_samd11_sorted = df_samd11.sort_values("preds")
df_samd11_inh = df_samd11_sorted.iloc[[0]]
df_samd11_inh["sequences"] = df_samd11_inh["utr100"].apply(lambda x: x[4:])
```

```
df_samd11_final = pd.concat([df_samd11_wt, df_samd11_inh])
df_samd11_final["len"] = df_samd11_final["sequences"].apply(lambda x: len(x))
df_samd11_final

# acot11, acot11,isg15
df_acot11_wt = df_acot11.iloc[[0]]
df_acot11_wt["sequences"] = df_acot11_final["utr100"].apply(lambda x: x[18:])

df_acot11_sorted = df_acot11.sort_values("preds")
df_acot11_inh = df_acot11_sorted.iloc[[0]]
df_acot11_inh["sequences"] = df_acot11_inh["utr100"].apply(lambda x: x[12:])

df_acot11_final = pd.concat([df_acot11_wt, df_acot11_inh])
df_acot11_final["len"] = df_acot11_final["sequences"].apply(lambda x: len(x))
df_acot11_final

# isg15, isg15,isg15
df_isg15_wt = df_isg15.iloc[[0]]
df_isg15_wt["sequences"] = df_isg15["utr100"].apply(lambda x: x[23:])

df_isg15_sorted = df_isg15.sort_values("preds")
df_isg15_inh = df_isg15_sorted.iloc[[0]]
df_isg15_inh["sequences"] = df_isg15_inh["utr100"].apply(lambda x: x[17:])

df_isg15_final = pd.concat([df_isg15_wt, df_isg15_inh])
df_isg15_final["len"] = df_isg15_final["sequences"].apply(lambda x: len(x))
df_isg15_final

df_mu_merged = pd.concat([df_samd11_final, df_acot11_final, df_isg15_final])
df_mu_merged

df_mu_merged["name"] = [
    "SAMD11_WT",
    "SAMD11_MIN_uORF",
    "ACOT11_WT",
    "ACOT11_MIN_uORF",
    "ISG15_WT",
    "ISG15_MIN_uORF",
]

df_merged_cs_final = df_merged_cs[["name", "preds", "sequences"]]
df_merged_mu_final = df_mu_merged[["name", "preds", "sequences"]]

df_ml_final = pd.concat([df_merged_cs_final, df_merged_mu_final])

df_ml_final.reset_index(drop=True, inplace=True)
df_ml_final.to_csv("df_ml_final.csv")

df_merged_cs.to_csv("df_merged_cs.csv")
```