UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Translational Control in Fungi

Permalink
https://escholarship.org/uc/item/78b422b4

Author
Clarkson, Bryan Kensey

Publication Date
2011

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Translational Control in Fungi

by

Bryan Kensey Clarkson

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Molecular and Cell Biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jennifer A. Doudna, Chair
Professor Jeremy Thorner
Professor Donald C. Rio
Professor Eva Harris

Spring 2011

Abstract

Translational Control in Fungi

by

Bryan Kensey Clarkson

Doctor of Philosophy in Molecular and Cell Biology

University of California, Berkeley

Professor Jennifer A. Doudna, Chair

Regulation of protein synthesis is critical for coordinating cellular activity with environmental cues. The extensive discrepancy between messenger RNA (mRNA) and protein levels exemplifies the pervasiveness of translational control. Accordingly, dysregulation of translation is associated with diseases, ranging from cancer to developmental, neuropsychiatric and metabolic disorders. Understanding cellular adaptation and the molecular underpinnings of diverse diseases requires a detailed understanding of translational control mechanisms. This work discusses the contributions of an essential translation factor to genome-wide protein synthesis and the ability of an unusual RNA structure element from Hepatitis C virus to promote translation in fungi.

Eukaryotic initiation factor 4G (eIF4G) is essential for protein synthesis. It simultaneously interacts with message- and ribosome-associated factors to help recruit and position the ribosome on the mRNA. The genomes of a wide variety of eukaryotes encode multiple eIF4G isoforms, some of which have unique translational functionality. To test the hypothesis that the two different individual eIF4G isoforms in yeast make distinct contributions to translation, I examined the effect of removing each eIF4G isoform on growth and translation in *Saccharomyces cerevisiae*. Strains expressing a single isoform at wild-type levels revealed that both isoforms have the capacity to support wild-type growth and translation rates. In fact, no messages expressed under standard laboratory conditions rely strongly on a particular eIF4G isoform for their translation. Environmental fluctuations affect the fitness of wild-type and isoform-specific strains equally, suggesting a large functional overlap exists between yeast eIF4G isoforms. Therefore, encoding eIF4G proteins from distinct genetic loci ensures maintenance of adequate eIF4G levels, which is critical for cellular fitness.

Several stresses reduce eIF4G levels to regulate translation. To determine whether there are message-specific effects of eIF4G reduction I compared the translational efficiency of all yeast mRNAs under wild type and eIF4G-limiting conditions. The translation of genes

that encode messages with longer poly-adenosine (poly(A)) tails are preferentially sensitive to eIF4G limitation. To test the hypothesis that translational enhancement by the poly(A) tail requires adequate eIF4G, I determined the effects of limiting eIF4G on the translation of mRNAs with varying poly(A) tail lengths. eIF4G limitation does not alter the relationship between poly(A) tail length and translation, *in vitro*. Therefore, mRNA-specific differences in the requirement for eIF4G are mediated by a poly(A)-independent mechanism.

The Hepatitis C virus (HCV) is a global public health threat and new therapeutics are needed. The development of new treatments requires a mechanistic understanding of critical stages in the viral life cycle. HCV utilizes an internal ribosome entry site (IRES) to initiate the synthesis of proteins required for viral replication. Extensive biochemical and structural studies have assigned functions to IRES components, however the influence of host factors on IRES activity remains obscure. To establish a genetic system for the study of host factor influence we determined the ability of the HCV IRES to initiate translation in *Schizosaccharomyces pombe*. The HCV IRES does not efficiently promote translation in *S. pombe in vivo* or *in vitro*. Initial steps have been taken to identify mammalian factors that are able to stimulate HCV IRES-mediated translation in *S. pombe*.

Acknowledgments

This work would not have been possible without assistance from several people. First and foremost I have to thank Jennifer Doudna for her unwavering support throughout my graduate career. I would not be where I am today without her persistence and boundless optimism. In addition to Jennifer, I have received extensive mentorship from Wendy Gilbert. Wendy's infectious excitement for science has inspired me on countless occasions. She introduced me to the project I focused on through most of graduate school and has been an endless source of encouragement. We have developed an amazing relationship both as colleagues as friends and I look forward to following the successes of the Gilbert lab in the coming years.

In addition to those who directly mentored me, the Doudna lab is full of amazing people that create an incredibly supportive place to do science. The breadth of knowledge in the lab is astounding and everyone is willing to offer his or her expertise on just about any stumbling block you may run into. The lab is also an extremely fun place to work and the constant joking is a nice balance to the stress of graduate school. I need to specifically acknowledge Blake Wiedenheft, who was always available to get outside and blow off some steam, whether it be for a run or a bike ride. It was also great to have a group of cyclists, specifically Blake, Andy Mehle and Scott Gradia, who were available to play hooky every now and then to take in the amazing riding in the area - whether it be in Berkeley's backyard on Thursday afternoons or exploring sections of the Northern California coast on midweek summer days.

In addition to relationships within the Doudna laboratory, I have benefited greatly from the scientific community at Berkeley. I would like to thank the members of my thesis committee, Jeremy Thorner, Don Rio and Eva Harris. Their insight throughout the years was instrumental in my growth as a scientist. Having been a rotation student in Jeremy's lab, he has always treated me as a lab member and been there for advice and guidance. I have benefited from Eva's expertise in translation and it is always fun to see a familiar face at the annual translation meeting. Don's knowledge of protein-RNA interactions has provided key insights at several stages of my graduate work. The UC Berkeley yeast community was extremely supportive and always available for advice, equipment and reagents. Help from the UC Berkeley Statistics department, specifically James Bullard, was critical when my project began to stray from the expertise of people in the Doudna lab. In addition to help from departments on the UC Berkeley campus, I received extensive assistance from the UCSCF community. Members of the Guthrie lab provided key insights at multiple stages of my graduate career. The UCSF Center for Advanced Technology (CAT) was an invaluable resource, providing equipment and expertise not available at Berkeley. I need to specifically acknowledge Adam Carroll, Paige Nittler and Katherine Hermes for their assistance.

In addition to it's rich scientific community, the Bay Area has been a phenomenal place to live. The natural beauty is breathtaking and ripe for exploring, whether it be on

bicycle or foot. I have met an amazing group of people here that will no doubt remain my lifelong friends. The city of Oakland has been a welcome escape from the UC-centric Berkeley community and served as a nice reset when school was getting to me. I have enjoyed following the success (and failure) of the Oakland Athletics and feel as though they are a critical element of the Oakland community and hope they remain that way - KEEP THE A'S IN OAKLAND!!! I also would like to thank Emily Johanna Warming for her support. She has been an amazingly understanding partner whose excitement and wonder are inspirational. I look forward to our future adventures.

Last but not least, I need to thank my family for their support and guidance. It has been great to have my sister, Elise, living in Oakland for the last 5 years. It is nice to have family so close to enjoy meals, dessert and baseball with. My Mom has been incredibly supportive in all of my endeavors and has helped immensely when times were tough. I deeply respect and do my best to emulate my Father. He is a very level headed and kind person whose perspective is always refreshing and helpful when things are grim. Mom, Dad and Elise, thank you for all of your support – Bryan

TABLE OF CONTENTS

Abstract 1
Acknowledgements i
Table of Contents iii
List of Figures vii
List of Tables viii
List of Symbols and Abbreviations ix

Chapter 1. Introduction 1
1.1 Biological significance of translational control 2
1.1.1 The economics of protein synthesis 2
1.1.2 Advantages of translational control 2
1.1.3 Roles of translational control 3
1.2 The process of translation 4
1.2.1 The stages of protein synthesis 4
1.2.2 Translation initiation is rate limiting 5
1.2.3 Translation initiation in the different domains of life 5
1.2.4 Eukaryotic translation initiation factors 9
1.3 Regulation of eukaryotic translation initiation 11
1.3.1 Global regulation of eukaryotic translation initiation 12
1.3.2 mRNA-specific regulation of eukaryotic translation initiation 13
1.4 Non-canonical translation initiation 15
1.4.1 Internal ribosome entry site (IRES)-mediated translation 15
1.5 Translation of Hepatitis C virus RNA 15
1.5.1 The Hepatitis C virus is a global public health problem 15
1.5.2 Mechanisms of Hepatitis C virus translation 16

Chapter 2. Functional overlap of *Saccharomyces cerevisiae* initiation factor 4G isoforms. 19
2.1 Introduction 20
2.2 Methods 21
2.2.1 Yeast strains and culture 21
2.2.2 Plasmid construction 22
2.2.3 Plasmid rescue 23
2.2.4 eIF4G ortholog detection 23
2.2.5 *Saccharomyces cerevisiae* eIF4G isoform sequence alignment 23
2.2.6 Doubling time determination 24
2.2.7 Preparation and fractionation of polysomes 24
2.2.7 Microarray analysis 25
2.2.7.1 Cell growth, lysis, and RNA isolation 25
2.2.7.2 cDNA synthesis and labeling, array fabrication, and hybridization. 25
2.2.7.3 Experimental design 25

2.2.7.4 Statistical analysis. 25
2.2.7.5 Raw data 26
2.2.8 Polyclonal antibody production 26
2.2.9 Immunoblot analysis. 26
2.2.10 Reverse-transcription quantitative PCR analysis (RT-qPCR) 27
2.3 Results 29
2.3.1 Diverse eukaryotes express multiple eIF4G isoforms 29
2.3.2 Deletion of *TIF4631* causes a translation initiation defect 30
2.3.3 *tif4631*Δ-associated phenotypes correlate with overall eIF4G levels 33
2.3.4 Yeast eIF4G isoforms have a similar capacity to support growth and translation. 34
2.3.5 The global translational profile of eIF4G isoform-specific strains is similar 36
2.4 Discussion 38

Chapter 3. Message-specific effects of limiting eIF4G availability 40
3.1 Introduction 41
3.2 Methods 42
3.2.1 Yeast strains and culture 42
3.2.2 Microarray analysis 42
3.2.2.1 Cell growth, lysis, and RNA isolation 42
3.2.2.2 cDNA synthesis and labeling, array fabrication, and hybridization 43
3.2.2.3 Experimental design 43
3.2.2.4 Statistical analysis 43
3.2.2.5 Raw data 43
3.2.3 Statistical analysis of modified array features 43
3.2.4 Reverse-transcription quantitative PCR analysis (RT-qPCR) 44
3.2.5 Construction of luciferase reporters with varying poly-adenosine tail length 44
3.2.6 *In vitro* translation 45
3.2.6.1 *In vitro* transcription of RNA templates 45
3.2.6.2 Translation extract preparation 45
3.2.6.3 Translation reaction and luciferase detection 45
3.2.7 RNA hybridization analysis of *in vitro* translated mRNA 46
3.3 Results 49
3.3.1 Depletion of eIF4G remodels the yeast transcriptome 49
3.3.2 *TIF4631*-deletion reduces translation of short mRNAs with long poly(A) tails 51
3.3.1 Translational efficiency correlates positively with poly(A) tail length *in vitro*. 56
3.4 Discussion 59

Chapter 4. Function of the Hepatitis C virus internal ribosome entry site in *Schizosaccharomyces pombe* 63
4.1 Introduction 64
4.2 Methods 65
4.2.1 Yeast strains and culture 65

4.2.2 Plasmid construction 65
4.2.3 GFP visualization 65
4.2.4 Immunoblot analysis 65
4.2.5 *In vitro* translation 66
4.2.5.1 *In vitro* transcription of RNA reporters 66
4.2.5.2 Extract preparation 66
4.2.5.3 *In vitro* translation and measurement of luciferase activity 66
4.3 Results and Discussion 69
4.3.1 *Schizosaccharomyces pombe* encodes a more mammalian-like eIF3 than *Saccharomyces cerevisiae* 69
4.3.2 The Hepatitis C virus IRES cannot efficiently initiate translation in *Schizosaccharomyces pombe*, *in vivo* 69
4.3.2 The Hepatitis C virus IRES cannot efficiently initiate translation in *Schizosaccharomyces pombe*, *in vitro* 72

Chapter 5. Summary 76
Appendix I. R code 79
I.1 *Saccharomyces cerevisiae* eIF4G isoform pairwise alignment 80
I.1.1 High Level Code 80
I.1.2 Functions 80
I.1.2.1 draw4GdomainMaps 80
I.1.2.2 alignmentHeatMap 81
I.2 Microarray Analysis 83
I.2.1 High Level Code 83
I.2.1.1 Ribosome Occupancy 83
I.2.1.2 Transcript Abundance 83
I.2.1.3 Polysome Abundance 84
I.2.2 Functions 84
I.2.2.1 doAnalysis 84
I.2.2.2 dwnWtCntrls 88
I.2.2.3 preProcess 89
I.2.2.4 twoSpot 90
I.2.2.5 corFit 91
I.2.2.6 linearModel 91
I.2.2.7 GOList 91
I.2.2.8 topTableList 92
I.2.2.9 writeTopTable 95
I.2.2.10 getMasterList 95
I.2.2.11 probeHomogenizeRG 97
I.2.2.12 probeHomogenizeMA 98
I.2.2.13 qualityPlots 99
I.2.2.14 y01eIF4GspotData 107
I.2.2.15 sigGenesPlot 112

1.2.2.16 vennDiagrams 112
1.2.2.17 MAplot 113
1.2.2.18 comPlot 114
1.2.2.19 makeBarPlot 115
1.3 Statistical analysis of message characteristics 115
1.4 *In vitro* translation data processing 130
1.4.1 High Level Code 131
1.4.2 Functions 131

Bibliography 144

LIST OF FIGURES

Figure 1.1: Correlation between yeast protein and message abundance **3**
Figure 1.2: The stages of protein synthesis **5**
Figure 1.3: Translation initiation in eukaryotes **8**
Figure 1.4: Translation initiation by Hepatitis C viral RNA **17**
Figure 2.1: Eukaryotic eIF4G isoform distribution **29**
Figure 2.2: eIF4G isoforms from *Saccharomyces cerevisiae* **30**
Figure 2.3: Polysome analysis of eIF4G deletion strains **31**
Figure 2.4: Growth and polysome analysis of strains expression exogenous Tif4631 ..... **32**
Figure 2.5: Doubling time, polysome and eIF4G protein level analysis in another commonly used strain background **33**
Figure 2.6: eIF4G transcript levels in different yeast strains **34**
Figure 2.7: Construction, eIF4G protein level, polysome and doubling time analysis of isoform specific strains **35**
Figure 2.8: Reproducibility of microarray datasets **36**
Figure 2.9: Differences in message-specific polysome abundance of wild type and isoform specific strains **37**
Figure 2.10: Summary of eIF4G protein level, translation initiation and growth rate data **39**
Figure 3.1: Effect of eIF4G gene deletions on genome-wide transcript abundance **49**
Figure 3.2: Reproducibility of transcript abundance microarray data sets **50**
Figure 3.3: Genome-wide analysis of ribosome occupancy in wild type and eIF4G deletion strains **53**
Figure 3.4: Reproducibility of ribosome occupancy data sets **54**
Figure 3.5: Microarray spot intensity and mRNA characteristic distributions of translationally modified genes **55**
Figure 3.6: Translation as a function of poly-adenosine tail length in wild type and eIF4G-limited extracts **58**
Figure 3.7: Global variation of message abundance and ribosome occupancy **60**
Figure 3.8: 5'-untranslated region analysis of translationally modified genes **61**
Figure 4.1: Comparison of conserved human and fission yeast eIF3 subunits **70**
Figure 4.2: Hepatitis C virus internal ribosome entry site function in *S. pombe in vivo* **72**
Figure 4.3: Hepatitis C virus internal ribosome entry site function in *S. pombe in vitro* **73**
Figure 4.4: The effect of mixing yeast and rabbit reticulocyte lysates on cap- and HCV IRES-dependent translation **75**

LIST OF TABLES

Table 2.1: Yeast strains used in this study 22
Table 2.2: Oligonucleotides used in this study 22
Table 3.1: Yeast strains used in this study 42
Table 3.2: Oligonucleotides used in this study 47
Table 3.3: Plasmids used in this study 48
Table 3.4: Characteristics of translationally modified features 56
Table 4.1: Oligonucleotides used in this study 68
Table 4.2: Plasmids used in this study 68
Table 4.3: Summary of eIF3 subunits from selected eukaryotes 70

LIST OF SYMBOLS AND ABBREVIATIONS

α	alpha
A site	amino acyl site of the ribosome
A260	absorbance at 260 nanometers
A600	absorbance at 600 nanometers
AA	amino acid
C-terminus	carboxy-terminus
CPE	cytoplasmic polyadenylation element
CrPV	cricket paralysis virus
cryo-EM	cryo-electron microscopy
Δ	gene deletion
DT	doubling time
E-MAP	epistatic miniarray profile
eIF	eukaryotic initiation factor
FMR	fragile x mental retardation
4E-BP	eIF4E binding protein
GAP	GTPase activating protein
GDP	guanosine 5'-di-phosphate
GEF	guanine nucleotide exchange factor
GTP	guanosine 5'-tri-phosphate
HCV	Hepatitis C virus
IF	prokaryotic initiation factor
IP	immunoprecipitation
IRES	internal ribosome entry site
kb	kilobase
LUC	luciferase
M	monosome
m7G	7-methyl guanosine
MAPK	mitogen-activated protein kinase
mDa	megadalton
Met-tRNAi(met)	initiator tRNA
MFC	multifactor complex
miR	microRNA
mRNA	messenger RNA
nm	nanometers
NT	nucleotide(s)
P	polysomes/polysomal RNA
P site	peptidyl transfer site
P/T	polysome/total = ribosome occupancy
PABP	poly(A) binding protein
PCR	polymerase chain reaction
PEG	polyethylene glycol

Φ	hydrophobic amino acid
Pi	inorganic phosphate
PI3K	phosphatidylinositol 3-kinase
PIC	preinitiation complex
PKR	protein kinase R
poly(A)	poly-adenosine
RBS	ribosome binding site
rDNA	ribosomal DNA
RF	release factor
Ribi	ribosome biogenesis
RNA	ribonucleic acid
RNP	ribonucleoprotein
RP	ribosomal protein
RRL	rabbit reticulocyte lysate
rRNA	ribosomal RNA
S	Svedberg units
SD	Shine-Dalgarno
SGA	synthetic genetic array
T	total RNA
TC	ternary complex
tRNA	transfer RNA
uORF	upstream open reading frame
UTR	untranslated region
YPAD	yeast extract plus peptone, adenine and dextrose

Chapter 1

Introduction

1. Introduction

1.1 Biological significance of translational control

1.1.1 The economics of protein synthesis

Proteins play key roles in nearly every biological process. They catalyze most cellular reactions and function as structural and signaling components. Accordingly, a large proportion of cellular resources are devoted to protein synthesis, also known as translation. For example, forty percent of essential genes in yeast encode components of the translational machinery [1]. The macromolecular complex responsible for polypeptide chain assembly, the ribosome, is a large (2.5 megaDaltons (mDa) in prokaryotes, >3 mDa in eukaryotes) ribonucleoprotein (RNP) that occupies 30-40% of the cytoplasmic volume of a rapidly growing yeast cell [2-4]. RNA components of the ribosome (rRNA) constitute 80% of cellular RNA and are encoded by a series of repeated genes that occupy 10% of the yeast genome [4]. In *Saccharomyces cerevisiae*, the ribosomal DNA (rDNA) is encoded at the *RDN1* locus, which is a ~1.5 megabase (Mb) region near the middle of the right arm of chromosome XII that consists of ~150 tandem copies of a 9.1 kb repeat. Within each repeat, there is a copy of the 5S (*RDN5*), 5.8S (*RDN58*), 25S (*RDN25*), and 18S (*RDN18*) genes, separated by three types of spacer regions - internal transcribed spacers (ITS1, ITS2), external transcribed spacers (5'-ETS, 3'-ETS), and non-transcribed spacers (NTS1, NTS2). The 5S rRNA is transcribed from each *RDN5* by RNA polymerase III, whereas the 5.8S, 18S, and 25S rRNAs are transcribed on the opposite strand from each *RDN18-RDN58-RDN25* repeat as a single precursor that includes ITS1 and ITS2 (35S pre-rRNA) by RNA polymerase I. Transcription of these three rRNA genes starts in 5'-ETS and terminates in the 3'-ETS. The resulting 35S pre-rRNA undergoes rapid endonucleolytic processing and extensive modification (e.g. 2'-O-methylation) in the nucleolus, concomitant with association of the resulting rRNAs with the appropriate ribosomal proteins. The 18S rRNA so generated is a component of the 40S ribosomal subunit, and the 5S, 5.8S and 25S rRNAs are components of the 60S subunit.

One-third of yeast messenger RNA (mRNA) encodes protein components of the ribosome, the processing of which comprises 90% of splicing events [4]. Additionally, the process of translation itself is energy intensive, hydrolysis of 4 high-energy bonds is required for each amino acid incorporated. In total, translation can consume up to 50% of energy generated by rapidly growing *Escherichia coli* [5]. Therefore, tight regulation of protein synthesis is essential for efficient energy budgeting.

1.1.2 Advantages of translational control

In addition to playing a key role in energy economics, the regulation of translation significantly influences the expression of most genes (Figure 1.1). Translational control is defined as a change in the number of protein molecules produced per mRNA per unit time. Affecting the efficiency with which pre-existing mRNA molecules produce proteins is an efficient and rapid way to control protein levels. Regulation of translation also affords precise temporal and spatial control, where the production of proteins can be limited to specific times and locations in the cell (reviewed in [6]). Translational control also enables

regulation under circumstances where other mechanisms are not possible. For example, some genes are so long (e.g. dystrophin, > 2000 kilobases (kb)) that their mRNA production and processing exceeds the timescale required for regulation. Additionally, systems where the levels of mRNA are not easily adjusted, such as in an embryo (where mRNA production is scarce) and mammalian red blood cells (which lack a nucleus), rely on translational control to regulate protein levels.

1.1.3 Roles of translational control

As a key step in gene expression, regulation of protein synthesis is critical for maintaining cellular homeostasis. Embryonic development, cellular differentiation and programmed cell death rely on the regulated production of numerous proteins (reviewed in [7-9]). Accordingly, errors in translational control are associated with a wide range of diseases, including cancer, neurological and metabolic disorders (reviewed in [10-12]). Translational control also plays a central role during stress response and viral infection, both from the perspective of host defense and viral evasion of immune defenses (Reviewed [13-15]). The extensive discrepancy between mRNA and protein levels in a wide range of organisms exemplifies the pervasiveness of translational control [16-19]. A recent study [19] compared the latest estimates of yeast mRNA and protein levels and observed a very poor correlation ($R^2 = 0.17$) (Figure 1.1).

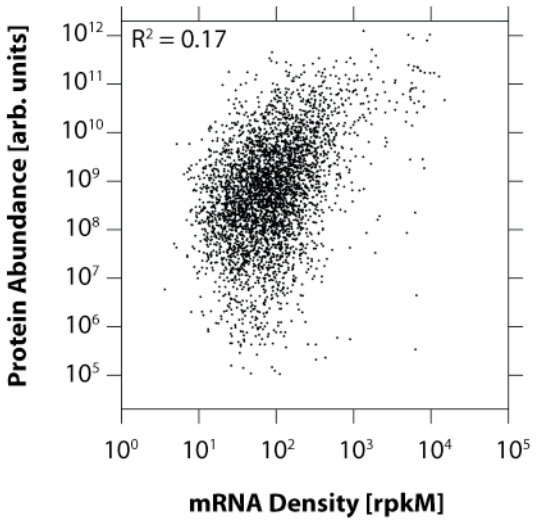


Figure 1.1. Correlation between yeast protein and message abundance. Protein (vertical) versus mRNA (horizontal) abundance. Relative protein levels were determined with stable-isotope labeling by amino acids in cell culture (SILAC)-based mass spectrometry [20] and relative mRNA levels were determined by deep sequencing of a complementary DNA (cDNA) library generated from fragmented total RNA [19]. The correlation (R^2) between these values is indicated. Figure adapted from S9 in [19].

Understanding cellular adaptation and the mechanisms underlying diverse diseases requires a detailed understanding of translational control phenomenon. The work described in this thesis addresses the contributions of an essential translation factor to genome-wide

protein synthesis and the mechanisms governing general translational regulation, as well as that of a specific viral mRNA. The significance of these studies requires an understanding of the current state of our knowledge about basic translation control mechanisms as well as those utilized by the Hepatitis C virus (HCV). An overview of these topics is provided in the subsequent sections of this Introduction.

1.2 The process of translation

1.2.1 The stages of protein synthesis

Conceptually, the process of protein synthesis is composed of four distinct steps: initiation, elongation, termination and recycling (Figure 1.2). During initiation, protein factors assemble and position a translation-competent ribosomal complex at the start codon of a mRNA (Reviewed in [21, 22]). Initiation is the target of the majority of translational control mechanisms and will be discussed in more detail below. Following initiation, the ribosome adds individual amino acids (AA) stepwise to the carboxy-terminus (C-terminus) of the growing polypeptide in a cycle of reactions called elongation (Reviewed in [2, 3]). During elongation, a site within the small ribosomal subunit (the aminoacyl- or A-site) facilitates base-pairing between the anti-codon stem of a transfer RNA (tRNA) and a triplet of nucleotides in the mRNA, which selects the proper tRNA-amino acid complex (known as aminoacyl-tRNA). When a cognate aminoacylated-tRNA is selected, RNA residues within the large subunit catalyze peptidyl transfer of the polypeptide chain onto the tRNA-associated amino acid. The ribosomal complex then translocates to vacate the aminoacyl-tRNA selection site, traversing the mRNA in a 5'- to 3'-direction. When the ribosomal complex and associated polypeptide encounter a stop codon this signals the termination of peptide synthesis. This recruits a factor (typically a proteinaceous release factor (RF)) that induces the hydrolysis and subsequent release of the completed protein (reviewed in [23]). Following termination intact ribosomal complexes are separated in a process known as recycling. This process regenerates individual subunits for a subsequent round of initiation. Recycling also releases the mRNA and final deacylated tRNA, however the molecular details of these reactions are still being elucidated (reviewed in [23]).

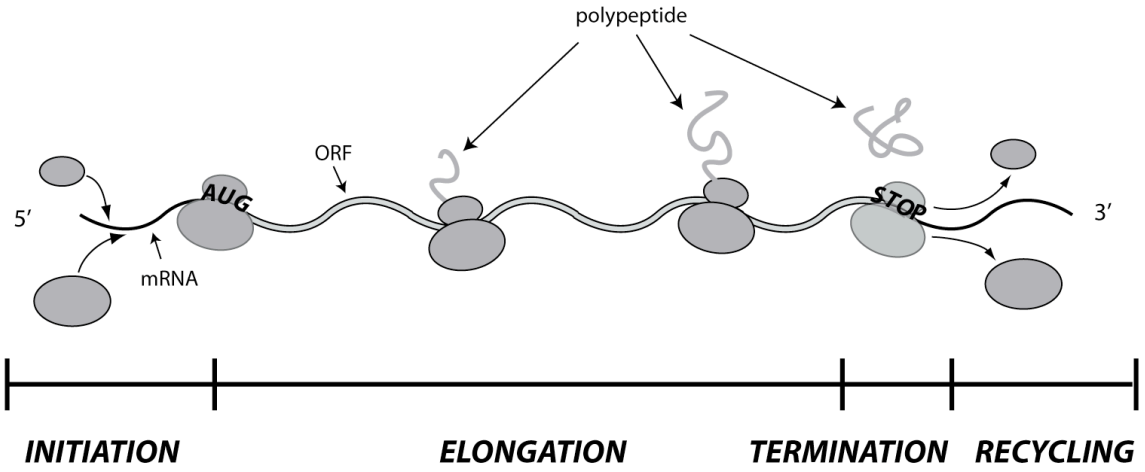


Figure 1.2. The stages of protein synthesis. Translation is composed of four basic steps: (1) Initiation, where ribosomal subunits are recruited in a stepwise fashion and positioned at the start codon (AUG) of a messenger RNA (mRNA; thin black line) open reading frame (ORF; thick gray line outlined in black), (2) Elongation, where a cycle of reactions adds amino acids individually to the growing polypeptide chain (thin gray line); (3) Termination, where a stop codon (STOP) is recognized by the elongating ribosome and the polypeptide is released and (4) Recycling, where ribosomal complexes are split to their constituent subunits and the mRNA is released in preparation for another round of translation. A more detailed description of each stage can be found in the text.

1.2.2 Translation initiation is rate limiting

It is generally believed that the number of protein chains initiated is the same number completed (i.e. that the ribosome will finish once it starts, even if it stalls) [1, 24, 25]. Therefore, in theory, the number of initiation events determines the amount of protein synthesized. Several lines of evidence suggest that initiation is the rate-limiting step of translation. First, the ribosome density (the number of ribosomes per open reading frame (ORF) length) is below the maximum packing capacity (e.g. based on the ribosomal footprint, translating mRNAs have the capacity to accommodate more ribosomes than they typically do) and is uniform (or slightly decreases) across messages [19, 26, 27]. In accordance with this, treatment of cells with translation elongation inhibitors does not have dramatic effects on protein synthesis rates [28, 29]. Second, rates of initiation (~6.5 seconds per event) exceed the time it takes to exit the initiation site (~2 seconds, based on elongation rates) [30]. Taken together, these studies suggest that initiation is the rate-limiting step of protein synthesis.

1.2.3 Translation initiation in the different domains of life

Given that initiation is the first and rate-limiting stage of translation, the majority of identified translational control mechanisms target this step. The mechanisms and molecular components of initiation vary between the three domains of life. However the process is similar in all cases: stepwise recruitment of ribosomal subunits to form a translation-competent complex positioned at the start codon of the mRNA. To recruit and position ribosomes, the majority of prokaryotic mRNAs contain a sequence (the Shine-Dalgarno

(SD) sequence) upstream of their protein-coding region that base pairs directly with a conserved RNA sequence in the 16S rRNA in the small ribosomal (30S) subunit [31, 32]. Subsequent association of the large subunit produces a translation-competent complex and protein synthesis can ensue. Three initiation factors (IFs) aid with different stages of this process in bacteria and their specific roles are the topic of several comprehensive reviews [2, 33]. The accessibility of the ribosome binding site (RBS), which contains the SD sequence and start codon, significantly influences the translational efficiency of prokaryotic mRNAs. Many of the characterized examples of prokaryotic translational control involve *trans*-acting factors (protein, RNA and metabolic intermediates) and *cis*-acting elements that modulate the availability of the RBS (Reviewed [34]).

The initiation process is less well understood in archaea. The early stages are likely to be similar to prokaryotes, since archaeal rRNAs/mRNAs contain SD sequences and direct interactions between mRNA and the ribosome have been detected [35]. Archaea contain more initiation factors that prokaryotes, but lack others that are critical for steps in the canonical eukaryotic initiation mechanism (e.g. scanning (see below)) (Reviewed in [36]). The roles of these additional initiation factors are the subject of ongoing research. Additionally, archaeal cells are enriched in a universally conserved class of mRNAs (leaderless) which lack SD-like sequences and therefore require a different mechanism of initiation which could be conserved in all domains of life [36]. The molecular mechanisms governing this type of initiation are actively being investigated [37].

Canonical initiation in eukaryotes can be divided into four steps: (1) Formation of the small ribosome containing 43S pre-initiation complex (PIC), (2) recruitment of the 43S PIC to mRNA to form the 48S complex, (3) scanning and start codon recognition by the 43S PIC, and (4) joining of the large subunit to form translation-competent 80S ribosomal complexes (Figure 1.3). This process involves a large number of proteins, referred to as eukaryotic initiation factors (eIFs), which are comprised of greater than 30 individual polypeptides. This process has been the subject of several comprehensive reviews and will be described briefly here [8, 21, 22]. The functions of individual initiation factors and regulatory mechanisms will be described in more detail in subsequent sections.

Initiation requires that post-termination 80S ribosomal complexes be split into 40S and 60S subunits in a process called recycling. The full complement of factors required to recycle eukaryotic ribosomes is still being elucidated, however it is clear that some initiation factors (eIFs 3, 1 and 1A) are involved (Reviewed in [23, 38]). Liberated small ribosomal subunits then associate with a number of factors (eIFs 1, 1A, 3, 5 and the "ternary complex" (TC), which is comprised of eIF2, GTP and the initiator tRNA (Met-tRNAi(met))) to form the 43S preinitiation complex (PIC). The order in which the 43S PIC assembles is not entirely clear and, in yeast, several factors (eIFs 1, 3, 5 and TC) pre-assemble into a multifactor complex (MFC) that stimulates 43S PIC assembly [22]. Once assembled, the 43S PIC is recruited to the 5'-end of most eukaryotic mRNAs via interactions with the message-associated eIF4F complex (containing eIFs 4A, 4B, 4E and 4G).

The majority of eukaryotic mRNAs are modified with a 7-methyl guanosine (m7G)[5']ppp[5']N (where N is any nucleotide) cap and poly-adenosine (poly(A)) tail at their 5'- and 3'-ends, respectively. Each feature interacts with a unique protein factor (the cap with eIF4E, and the poly(A) tail with poly(A)-binding protein (PABP)), both of which bind to a unique domain of the large scaffolding protein eIF4G. This mutually stabilizing network of interactions circularizes the message and allows distal mRNA features to influence 43S recruitment. eIF4G also stimulates the helicase activity of eIF4A and, via simultaneous interaction with eIF4E, targets it to the 5'-end of the message. This creates a conducive binding surface for the 43S PIC and "activates" the mRNA. Interactions between eIF4G and constituent factors, recruit the 43S complex to the cap proximal region of the mRNA, thus forming the 48S complex. eIFs 1, 1A, and 4G then stimulate the ribosomal complex to "scan" in the 5'- to 3'-direction. The eIF4A/B complex removes RNA secondary structures to facilitate scanning. During the scanning process eIF5, a GTPase-activating protein (GAP), stimulates GTP hydrolysis by eIF2 to aid in start site selection. When the scanning 43S complex encounters a start codon in a conducive sequence context, known as the Kozak consensus [39], base pairing between the mRNA AUG and the anticodon stem-loop of the Met-tRNAi(met), stimulates the release of eIF1 and inorganic phosphate (Pi) from eIF2-GDP. The GTPase eIF5B then catalyzes recruitment of the large, 60S, ribosomal subunit and stimulates the simultaneous release of additional initiation factors (eIFs 2, 3 and possibly eIF5). In the final step of initiation, eIF5B hydrolyzes its bound GTP in a ribosome-dependent manner. This reaction induces dissociation of both eIF5B and eIF1A and produces a translation-competent 80S ribosomal complex prepared to enter the elongation phase of protein synthesis.

As just described, the activities of translation initiation factors are required for efficient translation initiation. To further clarify their roles in the initiation process, the functions assigned to each factor are described in more detail in the following section.

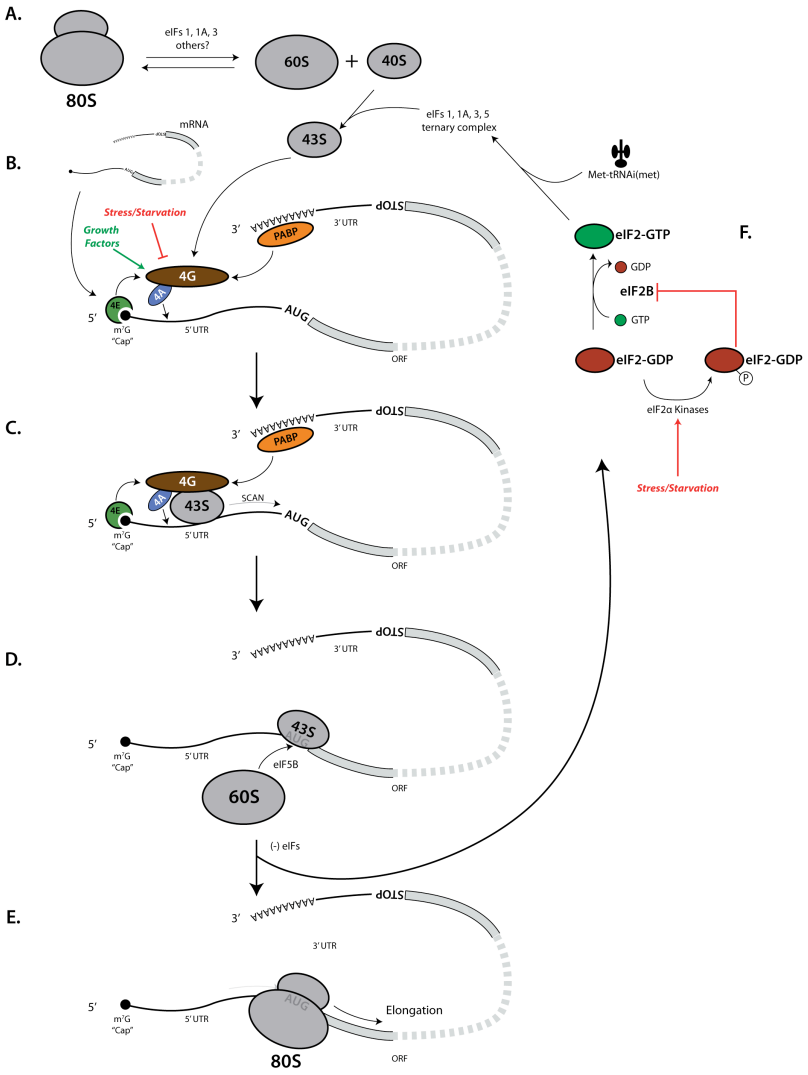


Figure 1.3. Translation initiation in eukaryotes. (**A**) To initiate protein synthesis post-termination 80S ribosomes must be separated into their constituent subunits by several eukaryotic initiation factors (eIFs). The small (40S) ribosomal subunit then associates with a number of factors (eIFs 1, 1A, 3, 5 and the ternary complex composed of eIF2, GTP, and initiator tRNA (Met-tRNAi(met)) to form a 43S complex. (**B**)The 43S complex is recruited to the 7-methyl guanosine (m7G) "cap" proximal region of the messenger RNA (mRNA) via a network of interactions nucleated by the large scaffolding protein eIF4G. (**C**) This complex then "scans" the 5'-untranslated region (UTR) of the message until it encounters the start (AUG) codon of the open reading frame (ORF). (**D**) Start codon recognition stimulates joining of the large (60S) subunit in a reaction catalyzed by eIF5B. (**E**) Upon 80S formation, eIFs are released and regenerated for subsequent rounds of initiation (F), and the 80S complex proceeds to the elongation phase of protein synthesis. The text contains a more detailed description of these events. The points at which some well characterized environmental cues (stress/starvation/ growth factors) inhibit (red) or stimulate (green) global translation rates are indicated.

1.2.5 Eukaryotic translation initiation factors

The process of eukaryotic translation initiation requires at least 12 initiation factors (eIFs) that are composed of greater than 30 polypeptides. Moreover, many of these factors have roles at several steps in the process. For clarity, the functions of each of these factors is briefly described below:

eIF1. eIF1 aids eIF3 and other factors in splitting 80S ribosomal complexes into 40S and 60S subunits [40, 41]. It promotes 43S preinitiation complex (PIC) assembly and, in yeast, is a member of the multifactor complex (MFC), also containing eIF3, eIF5 and the ternary complex (composed of eIF2, GTP, and initiator tRNA (Met-tRNAi(met))[22, 40, 42]. As a functional homolog of bacterial IF3, eIF1 plays key roles in start site selection [43]. It collaborates with eIF1A to stabilize the "open" form of the 40S subunit to promote scanning, and it's position in the 43S complex (near the peptidyl transfer site (P site) of the 40S) is critical for its gatekeeper function, which prevents release of Pi from eIF2-GDP-Pi until a proper start site is encountered [44-48]. Upon AUG recognition, it dissociates from the complex, allowing the final stages of initiation to ensue [49].

eIF1A: Like eIF1, eIF1A enhances the recycling of 40S and 60S from 80S complexes and promotes 43S complex assembly [40, 41]. As a structural and functional homolog of bacterial IF1, eIF1A is located in a position (the aminoacyl or A site) where it collaborates with eIF1 to induce a scanning competent conformation of the 40S ribosome [44-46]. During the scanning process its eukaryotic-specific C- and N-terminal extensions have opposite roles in AUG recognition, promoting continued scanning at non-AUG codons and halting scanning and promoting eIF1 release at cognate AUG codons, respectively [50].

eIF2: eIF2 is a heterotrimeric (alpha, beta and gamma subunits) GTPase that is associated with GTP and initiator tRNA (Met-tRNAi(met)) in the ternary complex (TC). As a member of this complex, eIF2 interacts specifically with and is responsible for delivering the correct initiator tRNA to the small (40S) ribosomal subunit [51]. The ternary complex has also been implicated in ribosome recycling and 43S assembly [21, 22, 41, 52]. Following recruitment of the 43S complex to mRNA, eIF2 is critical for proper start site selection because it anchors the initiator tRNA to the scanning 43S PIC until a proper AUG codon is encountered. During the scanning process eIF2 hydrolyzes its bound GTP in a reaction that requires eIF5, however release of inorganic phosphate is inhibited by eIF1 until the start codon is reached [48]. Upon start codon recognition and large subunit joining, the inactive eIF2-GDP complex dissociates. To regain affinity for initiator tRNA, active eIF2-GTP complex must then be regenerated in an eIF2B-catalyzed GTP- for GDP-exchange. This exchange is a key translational regulatory point and will be discussed in more detail below.

eIF2B: eIF2B is the guanine nucleotide exchange factor (GEF) for eIF2. It regenerates eIF2-GTP from the inactive eIF2-GDP complex that is released after each round of initiation (Reviewed in [22]). Several regulatory pathways modulate eIF2B activity to control ternary complex availability and regulate translation rates (see below) (Reviewed in [13, 53]).

eIF3: eIF3 is the largest initiation factor complex, composed of 13 subunits in humans (6 are conserved in *Saccharomyces cerevisiae*, 10 in *Schizosaccharomyces pombe*) each with unique biochemical properties and interacting partners. Accordingly, eIF3 stimulates most of the reactions in the initiation pathway (Reviewed in [38]). It is required to split post-termination 80S complexes into 40S and 60S subunits [40, 41]. eIF3 also stimulates ternary complex formation and helps assemble the 43S preinitiation complex through direct interactions with all PIC components, including the 40S ribosomal subunit ([54, 55] Reviewed in [22]). It aids in the recruitment of this complex to the mRNA via interactions with the mRNA and eIF4G (directly in mammals and via eIF1/eIF5 in yeast), and stimulates scanning of the mRNA for AUG recognition [56-60].

eIF4A: eIF4A is an ATP-dependent RNA helicase of the DExD/H-box family. Alone it is a weak, non-processive helicase and requires activation by eIF4B/eIF4H and eIF4G. The mechanism by which eIF4G stimulates the helicase activity of eIF4A is now understood in molecular terms and is thought to involve stabilization of the crucial DExD/H-box sequence motifs in a productive conformation [61]. eIF4A "activates" messages and aids in 43S PIC recruitment by melting cap proximal secondary structures to create a receptive binding surface [62]. Following recruitment, eIF4A facilitates scanning of the ribosomal complex by disrupting any secondary structures that might impede the process [63, 64]. This activity is required on most eukaryotic messages, since almost all have some secondary structure in their 5'-untranslated region (UTR) [45].

eIF4B+H: eIF4B and eIF4H are cofactors that stimulate the helicase activity of eIF4A. eIF4B is thought to increase the affinity of eIF4A for RNA, and since eIF4H exhibits significant homology to the N-terminus of eIF4B it is predicted to act in a similar, but less potent manner [65-67]. Consistent with its role in stimulating eIF4A helicase activity, yeast eIF4B displays many genetic interactions with eIF4A and its binding partners in the eIF4F complex (e.g. eIF4E and eIF4G) [68, 69]. Additionally, interactions have been detected between eIF4B and subunits of eIF3 in both yeast and mammals, suggesting it may play a role in 43S PIC recruitment to the mRNA [70, 71].

eIF4E: eIF4E interacts specifically with the m7G[5']ppp[5']N (where N is any nucleotide) "cap" moiety found at the 5'-end of all mRNAs transcribed by RNA polymerase-II [72-74]. Through its interaction with eIF4G, eIF4E stimulates 43S recruitment by targeting eIF4A helicase activity to the cap proximal region of the mRNA, which creates a accessible interaction surface. Also, via its interaction with eIF4G, eIF4E guides recruitment the 43S PIC to the 5'-end of the message, so that it can initiate scanning to the appropriate start codon. Interaction with a short conserved region of eIF4G (the $(I/X)_4L\mathbf{\Phi}$ motif, where **Φ** is any hydrophobic amino acid) confers most of the affinity between these two proteins, however these proteins have an extensive interaction surface and their association induces a conformational change in eIF4E that increases its affinity for the cap dramatically [75-79].

eIF4G: eIF4G is a large scaffolding protein that nucleates 48S complex formation via interactions with several initiation factors. Simultaneous interaction with eIF4E (the cap binding factor) and the poly(A) binding protein (PABP) circularizes the message and thereby allows features at distal mRNA ends to regulate ribosome recruitment [80-82]. This network of interactions is mutually stabilizing and increases the affinity of each protein for its substrate, which contributes to observed cap-poly(A) synergy ([76, 78, 79, 81-84]. The interaction between eIF4G and eIF4E is a major point of translational control and will be discussed in more detail below. Notably, recent studies found suggestive evidence that message circularization via the eIF4E-eIF4G-PABP complex is not essential in yeast and identified RNA-binding domains in eIF4G that functionally substitute for PABP binding [85]. eIF4G also stimulates eIF4A helicase activity and targets it to the 5'-end of the message via simultaneous interaction with eIF4E, the cap binding protein [61, 65, 66, 86-88]. This "activates" the mRNA by creating a receptive binding surface for the 43S PIC, which eIF4G then recruits to the message via interactions with associated factors (direct eIF3 contacts have been observed in mammals, whereas this interaction is mediated via eIF5 in yeast) [57-60]. Recent structural data suggest that eIF4G stimulates subsequent scanning of this complex by positioning eIF4A near the mRNA entry tunnel, allowing it to remove secondary structure in the 5'-untranslated region (UTR) that would impede the movement of the 43S toward the start codon [54, 64, 89]. Reportedly, certain yeast transcripts only require eIF4G at stages downstream of ribosome recruitment, suggesting that the role of eIF4G varies from message to message [90].

eIF5: eIF5 interacts with eIF2 and eIF3 in both yeast and mammals, and, as a member of the Multi-Factor Complex (MFC) in yeast, it has been implicated in 43S PIC assembly ([42, 91] Reviewed in [22]). eIF5 has also been shown to interact with eIF4G in yeast and is thought to play a role in recruitment of the 43S PIC to RNA [59, 60]. eIF5 functions as a GTPase-activating protein (GAP) and is required for GTP hydrolysis by eIF2 during the scanning process (Reviewed in [22]). eIF5 also interacts with eIF1A and influences ribosomal conformations required for proper start site selection [92].

eIF5B: eIF5B is a structural homolog of bacterial IF2. It is recruited to the 48S complex by eIF1A and catalyzes 60S joining in a ribosome-dependent manner ([93-95] Reviewed in [21]). Following 80S complex formation, eIF5B promotes release of initiation factors, including eIF1A, hydrolyzes GTP and dissociates, so that the first amnioacylated-tRNA can bind and protein synthesis can occur [96, 97].

1.3 Regulation of eukaryotic translation initiation

As the first and rate-limiting step of protein synthesis, most known translational control mechanisms target initiation. As described above, translation initiation is a complex, multi-step process with potential for regulation at each step. However, regulatory mechanisms commonly target two complexes: (1) the cap-binding eIF4F complex, which recruits the small ribosomal subunit to the 5'-end of mRNAs and (2) the ternary complex (composed of eIF2, GTP and initiator tRNA (Met-tRNAi(met)), which delivers the initiator

tRNA and its cognate N-terminal methionine to the small ribosomal subunit. The regulatory mechanisms targeting these two complexes can be grouped broadly into those that exhibit control globally or in a message-specific manner (for reviews see [8, 98, 99]).

1.3.1 Global regulation of eukaryotic translation initiation

Global translational control mechanisms typically modify the levels or activity of key initiation factors and therefore affect the translational efficiency of most cellular mRNAs. A well-characterized and conserved global translational control mechanism regulates availability of the ternary complex, composed of eIF2, GTP and initiator tRNA (Met-tRNAi(met)) (Reviewed in [13, 22, 53, 100]). As mentioned above, the eIF2-GTP complex delivers the initiator tRNA to the small ribosomal subunit. During initiation, proper start site selection relies on GTP hydrolysis by eIF2, which promotes its disassociation from the ribosomal complex and allows entry into the elongation phase. Following release, eIF2 must exchange GDP for GTP to regain affinity for initiator tRNA and regenerate ternary complex for subsequent rounds of initiation. eIF2 cannot readily exchange GTP for GDP itself; instead, it requires the guanine nucleotide exchange factor (GEF) eIF2B to catalyze the swap. In all cells examined eIF2B is limiting with respect to eIF2 and therefore must act catalytically. A number of cellular stresses activate kinases that phosphorylate eIF2 (at serine 51 of the mature alpha subunit), converting it into a competitive inhibitor of eIF2B by increasing its affinity. This titrates out free eIF2B, inhibiting GDP-GTP exchange on the remaining eIF2 molecules, thus blocking ternary complex regeneration and reducing global translation initiation rates.

In addition to availability of the ternary complex, the levels of several initiation factors are reduced by site-specific cleavage during times of stress. For example, apoptosis-specific cysteine proteases, called caspases, cleave several initiation factors, including eIF2, eIF4B, eIF3 and eIF4G to rapidly inhibit protein synthesis. The contribution of these cleavage events to translational inhibition is still being elucidated (For review see [9]). A number of virally-encoded proteases specifically cleave eIF4G, separating the domains required for interaction with the cap- and poly(A)-binding proteins from those that mediate ribosome recruitment. This simultaneously inhibits the translation of host messages, which rely on intact eIF4G for ribosome recruitment (see above), and activates the translation of viral mRNAs, which contain special RNA sequences that functionally substitute for the interaction between eIF4G and message-associated factors (see below and for reviews see [14, 15]).

In contrast to widespread downregulation of translation by stress, growth stimulatory signals enhance global translation. A number of extracellular cues enhance translation rates by regulating the availability of the cap-binding protein, eIF4E. As mentioned above, eIF4E guides recruitment of the small ribosomal subunit to the 5'-end of mRNAs via simultaneous interactions with the 7-methyl guanosine cap and the large scaffolding protein eIF4G. Interactions between eIF4E and eIF4G require a small domain of eIF4G that is shared by another family of proteins known as the 4E-binding proteins (4E-

BPs). 4E-BPs competitively inhibit the eIF4E-4G interaction to negatively regulate translation. However, 4E-BPs only tightly associate with eIF4E in their hypophosphorylated state. Growth stimulatory signals, such as insulin, activate a number of kinases that phosphorylate and reduce the affinity of 4E-BPs for eIF4E, thereby enhancing global translation. The tumor suppressor programmed cell death protein 4 (PDCD4) is another phospho-protein that was recently found to regulate protein synthesis [REF: Dorrello et al Science 2006]. In the absence of growth factors, PDCD4 binds and inhibits the helicase activity of eIF4A. However, in response to growth stimulatory signals, kinases phosphorylate PDCD4 on a specific residue and target it for ubiquitin-mediated degradation, thus allowing efficient protein synthesis. In addition to phosphorylating regulatory proteins, a number of growth factor stimulated pathways, including the mammalian target of rapamycin (mTOR)/phosphatidylinositol 3-kinase (PI3K)/Akt and Rat Sarcoma (Ras)/Mitogen-activated protein kinase (MAPK) signaling modules, directly phosphorylate initiation factors like eIF4E, eIF4G and eIF4B to stimulate translation (for reviews see [8, 101]).

1.3.2 mRNA-specific regulation of eukaryotic translation initiation

As the name implies, mRNA-specific control regulates the expression of particular messages. Many examples of message-specific control involve activation or repression of translation by *trans*-acting protein factors that interact with specific *cis* regulatory elements in the 5'- and 3'-untranslated regions of target mRNAs (reviewed in [99]). For example, a number of genes involved in iron metabolism contain a cap-proximal stem loop structure, known as the iron-responsive element (IRE), that is bound by the iron regulatory proteins (IRP) under conditions of low iron. This interaction does not inhibit binding of the cap-binding complex (eIF4F), but sterically hinders ribosome recruitment by eIF4F and therefore inhibits the translation of these genes.

During oocyte maturation in *Xenopus*, the translation of maternally deposited mRNAs is activated by cytoplasmic polyadenylation. A U-rich sequence in the 3'-UTRs of target genes, known as the cytoplasmic polyadenylation element (CPE), guides polyadenylation. To prevent premature translation of these messages, cytoplasmic polyadenylation element-binding protein (CPEB) binds the CPE and recruits Maskin, which competitively inhibits the interaction between the cap-binding protein (eIF4E) and eIF4G. Analogous mechanisms ensure proper spatiotemporal production of proteins required for mating-type switching in budding yeast, embryogenesis in *Drosophilla* and *Xenopus* and neuronal function in metazoans (for review see [6]).

In addition to binding sites for specific protein factors, many mRNAs contain sequences that interact directly with the translation machinery to influence protein synthesis. For example, upstream open reading frames (uORFs) are short open reading frames found in the 5'-untranslated region of approximately 20-30% of yeast and mammalian genes ([19, 102]). uORFs typically act as translational repressors, since reinitiation at the downstream *bona fide* ORF is generally inefficient, however, in some

cases the configuration of uORFs can stimulate translation under limiting ternary complex conditions [99, 103]. The yeast *GCN4* transcript is the most well-characterized example of this phenomenon. *GCN4* encodes a transcription factor that enhances the expression of several amino acid biosynthetic genes. Amino acid starvation activates Gcn2, the sole yeast eIF2α kinase, and reduces ternary complex levels (see above). Under these conditions translation of most transcripts is reduced; however the configuration of the four uORFs in the *GCN4* mRNA permits read-through to the protein-coding ORF, ultimately stimulating biosynthesis of the required amino acids. A number of groups have utilized the powerful *GCN4*-based genetic system, first established by Alan Hinnebusch and co-workers, to elucidate mechanistic details of several steps in translation initiation including scanning and start codon recognition (see [100]).

Another class of RNA elements that influence translation initiation are internal ribosome entry sites (IRESs). As the name implies, an IRES interacts with the translation machinery to recruit ribosomes in cap-independent manner. This strategy of ribosome recruitment was first identified in certain viral RNAs, but also seems to be an important mechanism that allows for the translation of particular host messages during times of widespread translational inhibition, such as during nutrient limitation ([104] reviewed in [105]). This mode of translation initiation is discussed in more detail in the next section.

Regulation of translation by microRNAs represents an emerging example of message-specific translational control. MicroRNAs (miRNAs) are small (~20-30NT) non-coding RNAs that extensively regulate gene expression post-transcriptionally both by influencing the stability and translation of mRNAs. miRNAs act as "guides" that direct effector proteins to bind to messages via Watson-Crick base pairing. miRNA binding sites are often found in the 3'-UTR of a target transcript. In general, the extent of base pairing influences whether regulation will occur via target cleavage or translational modulation. miRNAs target both translation initiation and elongation, however the mechanisms by which this regulation occurs are still not clear. Three mechanisms have been proposed for inhibition of initiation: (1) competition between the miRNA complex and eIF4E for cap binding, (2) miRNA-complex induced deadenylation and (3) blockage of large subunit joining by the miRNA complex. More experiments are needed to determine the contribution of these mechanisms as well as ribosome drop off during elongation to miRNA-induced gene regulation (for review see [8, 106]).

This listing is by no means a comprehensive coverage of translational control mechanisms. In theory, every component involved in protein synthesis can be regulated to influence global or message-specific translational efficiency. The recent advent of short-read sequence-based tools to measure genome-wide translational efficiency with high resolution will no doubt continue to identify novel examples of translational control that will keep biochemists and geneticists busy for quite some time [19].

1.4 Non-canonical translation initiation

As described in previous sections, most eukaryotic mRNAs utilize a cap structure and several intiation factors to assemble and correctly position a translation-competent ribosomal complex for protein synthesis. In contrast, a number of messages utilize alternative strategies for ribosome recruitment positioning and translation, such as end dependent initiation (where ribosomes are loaded onto the 5'-end of an message in the absence of many initiation factors), ribosome shunting (where the small ribosomal subunit can skip large regions during the scanning process), leaky scanning (where the 5'-most AUG is bypassed and a downstream AUG is used for initiation), non-AUG initiation (where protein synthesis initiates at a non-AUG codon), frameshifting (where the ribosome shifts its reading frame during protein synthesis to increase the coding capacity of certain sequences) and readthrough (where the stop codon is ignored to produce an extended polypeptide) (reviewed in [14]).

Another class of messages, first identified in viral RNAs, contain sequences in their 5'-untranslated regions (UTRs) that bypass the requirement for the cap and many, and in extreme cases all, initiation factors. These sequence elements generally recruit ribosomes "internally" (i.e. distal to the 5'-end of the transcript) and therefore are referred to as Internal Ribosome Entry Sites (IRESs).

1.4.1 Internal ribosome entry site (IRES)-mediated translation initiation

The first IRES identified was in the poliovirus mRNA. Since this RNA was known not to be capped, it was speculated to be translated in a cap-independent manner [107]. Subsequent experiments identified an element of the poliovirus RNA that could mediate initiation of the downstream cistron of a polycistronic message and on circular RNAs, definitively showing that ribosomes were being recruited in an end-independent manner [108, 109]. A groundbreaking set of biochemical experiments identified the minimal set of initiation factors required for a related IRES from encephalomyocarditis virus (EMCV) [110, 111]. *Trans*-acting protein factors (called ITAFs) also influence the translation of particular IRESs, however the influence of ITAFs and the dependence on initiation factors varies significantly from IRES to IRES. A discussion of these differences is beyond the scope of this discussion, but is the subject of comprehensive reviews (e.g. [15]). Since a section of this thesis focuses on investigating HCV IRES function, its mechanism of translation initiation is discussed further below.

1.5 Translation of hepatitis C virus RNA

1.5.1 The Hepatitis C virus is a global public health problem

Hepatitis C virus infects >180,000,000 worldwide and is the causative agent of chronic liver diseases, including cirrhosis and hepatocellular carcinoma [112]. HCV infection is extremely persistent, with greater than 80% of infected individuals unable to eliminate the virus and at high risk of developing chronic liver diseases [113]. New therapeutic strategies are needed, since the current combination of PEG-ylated interferon-alpha and Ribavirin used is ineffective against certain HCV genotypes and is associated with severe side effects [114]. A more detailed mechanistic understanding of HCV translation will aid in the design

of more effective treatments. The current model for HCV IRES-mediated translation is discussed below.

1.5.2 Mechanisms of Hepatitis C virus translation

HCV is a positive strand RNA flavivirus with a genome of ~9.6 kilobases (kb). The genome contains a single open reading frame that encodes a polyprotein of slightly greater than 3000 amino acids, which is processed by a combination of viral and host proteases to yield at least 10 viral proteins (reviewed in [115]) (Figure 1.4A). There are 6 major HCV genotypes worldwide that differ by up to 30% in sequence; however the 5'- and 3'-untranslated regions exhibit high sequence conservation [115]. Sequences in the 3'-UTR have been implicated in viral replication and translation initiation, and the 5'-UTR contains two microRNA-122 binding sites that regulate viral RNA abundance in liver cells [116-120]. However, the majority of the 5'-UTR sequence is dedicated to an elaborate internal ribosome entry site that mediates cap-independent, internal initiation of HCV translation [121, 122].

Mutagenesis and chemical modification studies have mapped five domains of significant secondary structure in the HCV RNA 5'-UTR [123-126] (reviewed in [15, 127, 128]) (Figure 1.4B). Domains II-IV constitute the IRES, and its function relies on their structural integrity. In combination, the domains combine to adopt an ion-dependent 3D fold that binds directly to 40S ribosomal subunits [115, 129-138]. This was the first example of a eukaryotic mRNA recruiting the ribosome directly, a property that was thought to be specific to prokaryotic and archaeal messages [136]. Additionally, the IRES recruits the ribosome directly to the initiation codon, bypassing the requirement for scanning that exists in most other eukaryotic messages [125, 139-141]. The 40S interaction surface is substantial, encompassing the junctions between several domains and this interaction induces a conformational change in the 40S subunit that opens up the mRNA entry channel and facilitates initiation [134, 136, 138, 141-144] (Figure 1.4C).

However, the ability of the HCV IRES to recruit 40S subunits is not sufficient for productive translation. IRES mutants that maintain efficient 40S binding, but are defective in translation, have been isolated and initial characterization found that the IRES also requires eIF2 to deliver and correctly position the Met-tRNAi(met) on the 40S, eIF3 to help stabilize this complex, and eIF5/5B to catalyze 60S joining [123, 130, 134, 136, 145]. It has recently been appreciated that the IRES may also initiate translation in an eIF2-independent manner that could predominate under conditions of increased eIF2α phosphorylation, which occurs via increased protein kinase R (PKR) activity induced by HCV infection [146, 147] [148]. eIF3 interacts directly and specifically with the IRES, and initial biochemical characterization of the IRES-eIF3 interaction identified an extensive interface, with several of the subunits in close proximity to RNA (see magenta box in Figure 1.4B) [134, 137, 138, 149]. Single particle cryo-electron microscopy (cryo-EM) expanded upon the intricacies of this interaction, providing the first look at the extensive intertwining of these two molecules [54]. Fortuitously, the IRES adopts a very similar structure when complexed with the 40S

subunit and eIF3. This allowed modeling of eIF3 and other initiation factors onto the 40S and provided insights into mechanisms of canonical translation initiation.

Order-of-addition studies suggest that the IRES first associates with the 40S subunit and then recruits eIF2-Met-tRNAi(met) and eIF3 [140, 145, 150] (Figure 1.4C). These studies, in combination with extensive biochemical characterization, have assigned functions to different IRES domains. Domain II has been implicated in 40S subunit recruitment and is positioned in the tRNA exit site where it induces conformational changes that facilitate eIF2 release, eIF5-mediated GTP hydrolysis and 60S joining [138, 141, 144, 151]. The basal portion of domain III is required for 40S recruitment, while the apical portion contributes IRES affinity for eIF3 [137, 138]. Additionally, the proposed pseudoknot that forms adjacent to domain IV has recently been shown to position the initiation codon in the 40S subunit for translation initiation [152].

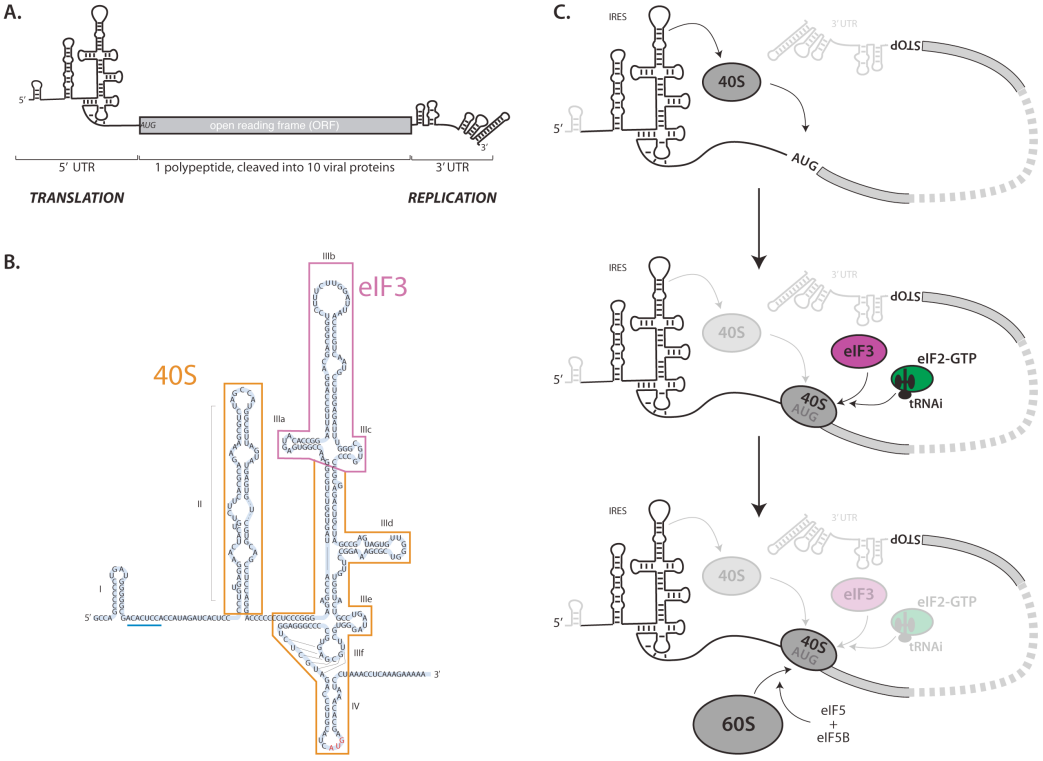


Figure 1.4. Translation initiation by Hepatitis C viral RNA. (**A)** A schematic of the HCV RNA, with the 5'- and 3'-untranslated regions (UTRs) and the single open reading frame indicated. (**B)** A detailed secondary structure prediction of HCV 5'-UTR. Functional domains are labeled (I-IV), with the subdomains of domain III also indicated (a-f). The nucleotides implicated in 40S and eIF3 binding are boxed in gold and magenta, respectfully. This figure is adapted from Fraser and Doudna Nat Rev Micro 2007. (**C)** The hepatitis C viral (HCV) RNA recruits ribosomes in a non-canonical fashion, utilizing a structured element (known as an internal ribosome entry site (IRES) in its 5'-untranslated region (UTR) to bind directly to the small (40S) ribosomal subunit and correctly position it at the start (AUG) codon in a cap-independent manner. This complex then recruits the ternary complex (composed of eIF2, GTP and initiator tRNA (Met-tRNAi(met)), whose association is stabilized by eIF3, which is recruited via direct interactions with the IRES element. Large subunit (60S) joining and subsequent entry into the elongation phase then proceed in a manner similar to that of most host messages.

Despite extensive biochemical and structural studies, the lack of a tractable genetic system has hindered identification and examination of host factors influencing HCV IRES function. Several attempts have been made to utilize *Saccharomyces cerevisiae* for this purpose; however, the ability of the HCV IRES to mediate translation in this organism is unclear. One section of this work describes the attempts to get the HCV IRES to promote translation in *Schizosaccharomyces pombe*, which has a more mammalian-like translation system than *S. cerevisiae.* Establishment of a genetic system to study HCV function would enable further elucidation of HCV translation, facilitate the development of more effective treatments and shed light on fundamental initiation mechanisms.

Chapter 2

Functional overlap of *Saccharomyces cerevisiae* initiation factor 4G isoforms*

* The work presented in this chapter has previously been published as part of the following paper: Clarkson BK, Gilbert WG, Dounda JA. (2010) Functional overlap between eIF4G isoforms in *Saccharomyces cerevisiae*. *PLoS ONE* 5: (2) pp. e9114.1-e9114.15.

Bryan K Clarkson performed the experiments and analyzed the data. Wendy V Gilbert and Jennifer A Doudna assisted with experimental design and manuscript preparation.

2. Functional overlap of *Saccharomyces cerevisiae* initiation factor 4G isoforms

2.1 Introduction

Translation initiation is the rate-limiting step of protein synthesis in which the ribosomal subunits assemble with initiation factors and an mRNA to form an activated complex (see Chapter 1; reviewed in [21, 22]). eukaryotic initiation factor 4G (eIF4G) is central to this process because it simultaneously recognizes message- and ribosome-associated proteins, thereby nucleating the ribosome-mRNA interaction [62]. In canonical, cap-dependent initiation, an mRNA is selected for translation via interactions of its 5'-methyl-7-guanosine (m7G) cap and 3'-poly-adenosine tail with the cap (eIF4E) and poly(A) (PABP) binding proteins, respectively (reviewed in [21, 22]). Recognition by eIF4G enhances the affinity of these two factors for their substrates [76, 79, 83]. The mutually stabilizing interaction of these three proteins circularizes the message and facilitates regulation of translation by distal mRNA elements [80-82]. Following message selection, eIF4A, an RNA helicase whose activity is stimulated and targeted by direct interaction with eIF4G, destabilizes secondary structure in the 5'-end of the message, favoring small (40S) ribosomal subunit association [65, 66, 86, 87]. eIF4G recruits the 40S subunit to the message via simultaneous binding of eIF4E, PABP, the mRNA and factors connected to the 40S [57-59]. The 40S-containing complex then "scans" to the initiation codon, the large (60S) subunit joins and protein production ensues. In addition to its role in canonical translation initiation, eIF4G is also required for 5'-cap-independent translation of some host and viral mRNAs [104, 153, 154].

Interestingly, the genomes of diverse eukaryotes encode more than one form of eIF4G. Previous work suggests that eIF4G variants make different contributions to cellular translation. In wheat (*Triticum aestivum*) and yeast (*Saccharomyces cerevisiae*), one eIF4G isoform is better at promoting the translation of cap-independent messages *in vitro* [155, 156]. Recently identified eIF4G isoforms in both *Drosophila melanogaster* and *Caenorhabditis elegans* predominate in germline cells and are essential for timely translation of genes important for development [157-159]. Following infection, some viruses induce cleavage of eIF4G, creating eIF4G truncation variants that simultaneously promote viral and inhibit host translation (reviewed in [160, 161]). Depletion of particular mammalian eIF4G isoforms, via selective cleavage or siRNA-mediated silencing, impairs the translation of distinct subsets of host mRNAs [162-164].

To test the hypothesis that individual eIF4G isoforms stimulate translation of distinct sets of genes in yeast, we examined the effect of deleting each eIF4G isoform, encoded by *TIF4631* and *TIF4632*, on growth and translation in *S. cerevisiae*. During exponential growth in rich media, deletion of *TIF4631* reduces proliferation and global translation rates. However, total eIF4G levels are significantly reduced in *tif4631*Δ stains and the severity of *tif4631*Δ-associated phenotypes correlates with eIF4G abundance. The creation of strains containing wild type levels of a single eIF4G variant ("homogenic" strains) revealed that both isoforms have a similar capacity to support growth and translation initiation rates. Moreover, these strains translate a wild type complement of messages, suggesting that no

messages expressed during log phase growth in rich media rely strongly on a specific eIF4G isoform for their translation. Taken together, these results reveal the large functional overlap between eIF4G isoforms in yeast.

2.2 Methods

2.2.1 Yeast strains and culture

All yeast cultures are S288C derived. YBC86 is BY4742 (*MAT*alpha *his3*Δ*1 leu2*Δ*0 lys2*Δ*0 MET15 ura3*Δ*0;* [165] and YBC87 is from [166] (*MATa his3 trp1 ura3* L-A-o M-o Gal+). All various eIF4G mutants were created from one of these two parents (see Table 2.1). Yeast cultures were grown in YPAD (1% Yeast extract, 2% Peptone, 0.01% Adenine hemisulfate, 2% Dextrose) at 30°C according to standard protocols [167], unless otherwise noted.

Homogenic strains were created using a two-step (selection, counter-selection) gene replacement method [167]. The entire *TIF4631 or TIF4632* ORF (ATG to STOP codon) was replaced with the *URA3* marker using PCR-based gene manipulation methods [168]. Specifically, *URA3* was amplified from pRS306 with forward (5') primers that add 40NT of genomic sequence immediately upstream (-40 to -1) of the *TIF4631 or TIF4632* start codon (*TIF4631*: AAGCAAGGTAAGAGGACAACTGTAATTACCTATTACAATAga ttcggtaatctccgaaca; *TIF4632*: AGAATTTATTACAAAGAACAATAGATCAATTGTAGGCA CTgattcggtaatctccgaaca; the eIF4G sequence in UPPERCASE and the *URA3* sequence is in lowercase) and reverse (3') primers that add 40NT of genomic sequence immediately downstream of the stop codon of *TIF4631 or TIF4632* (+1 (after stop) to +40; *TIF4631*: TCCAAGTGACATTTTCGATACTTAACATGATCTATTCATGcacaccgcag ggtaata act; *TIF4632*: AAGGAAAAAGACTAGCTTATCGTTTCTAAAAGAAAATCTTcacaccgcagg gtaata act; the eIF4G sequence in UPPERCASE and the *URA3* sequence is in lowercase). The PCR product was transformed using standard methods [169] and *tif4631Δ or tif4632*Δ strains were selected for on Synthetic Complete (SC) (-ura) and confirmed by PCR with forward primers ~200bp upstream of the replaced ORF (*TIF4631*: CCGCGTTCTCTGTG TGCAACGGATG; *TIF4632*: GAGGTAAACACAGCAAACGACCAC) and reverse primers within the *URA3* marker sequence (GCTTGGCAGCAACAGGACTAGGATG). The *URA3* marker was then replaced with *TIF4632 or TIF4631* using an identical approach (*TIF4631* locus forward: AAGCAAGGTAAGAGGACAACTGTAATTACCTATTACAA TAatgactgaccaaagaggtcc; reverse: TCCAAGTGAATTTTCGATACTTAACATGATCTAT TCATGttaatcactgtccccatcg; *TIF4632* locus forward: AGAATTTATTACAAAGAACAATAG ATCAATTGTAGGCACTatgacagacgaaactgctc; reverse: AGGAAAAAGACTAGCTTATCG TTTCTAAAAGAAAATCTTtactcttcgtcatcactttc; targeting sequence is in UPPERCASE and replacement sequence is in lowercase) except selection of positive transformants was carried out on 5-fluoroorotic acid (5-FOA). Precise gene replacement was confirmed by sequencing.

Table 2.1. Yeast Strains used in this study.

Strain	Genotype	Source
YBC28	*MATa his3 tr p1 ura3* L-A-o M-o Gal+ *tif4632::TIF4631*	This study
YBC29	*MAT*alpha *his3Δ1 leu2Δ0 lys2Δ0 MET15 ura3Δ0 tif4632::TIF4631*	This study
YBC30	*MATa his3 tr p1 ura3* L-A-o M-o Gal+ *tif4631::TIF4632*	This study
YBC31	*MAT*alpha *his3Δ1 leu2Δ0 lys2Δ0 MET15 ura3Δ0 tif4632::TIF4631*	This study
YBC86	*MAT*alpha *his3Δ1 leu2Δ0 lys2Δ0 MET15 ura3Δ0*	[170]
YBC87	*MATa his3 tr p1 ura3* L-A-o M-o Gal+	[166]
YBC88	*MATa his3 tr p1 ura3* L-A-o M-o Gal+ *tif4631::HIS5+(Sz. Pombe)*	This study
YBC89	*MATa his3 tr p1 ura3* L-A-o M-o Gal+ *tif4632::HIS5+(Sz. Pombe)*	This study
YBC90	*MAT*alpha *his3Δ1 leu2Δ0 lys2Δ0 MET15 ura3Δ0 tif4631::kan*	This study
YBC91	*MAT*alpha *his3Δ1 leu2Δ0 lys2Δ0 MET15 ura3Δ0 tif4632::kan*	This study

Table 2.2. Oligonucleotides used in this study.

Ref	Name	Sequence
OBC53	TIF4631::URA_F	AAGCAAGGTAAGAGGACAACTGTAATTACCTATTAC AATAGATTCGGTAATCTCCGAACA
OBC54	TIF4631::URA_R	TCCAAGTGACATTTTCGATACTTAACATGATCTATTC ATGCACACCGCAGGGTAATAACT
OBC55	URA::TIF4632_F	AAGCAAGGTAAGAGGACAACTGTAATTACCTATTAC AATAATGACTGACCAAAGAGGTCC
OBC56	URA::TIF4632_R	TCCAAGTGACATTTTCGATACTTAACATGATCTATTC ATGTTAATCACTGTCCCCATCG
OBC57	TIF4632::URA_F	AGAATTTATTACAAAGAACAATAGATCAATTGTAGG CACTGATTCGGTAATCTCCGAACA
OBC58	TIF4632::URA_R	AAGGAAAAAGACTAGCTTATCGTTTCTAAAAGAAAA TCTTCACACCGCAGGGTAATAACT
OBC59	URA::TIF4631_F	AGAATTTATTACAAAGAACAATAGATCAATTGTAGG CACTATGACAGACGAAACTGCTC
OBC60	URA::TIF4631_R	AGGAAAAAGACTAGCTTATCGTTTCTAAAAGAAAAT CTTTTACTCTTCGTCATCACTTTC
OBC5	TIF4631_BamHI	GCAGGGATCCATGACAGACGAAACTGCTCACCCGA CACAA
OBC6	TIF4632_BamHI	GCAGGGATCCATGACTGACCAAAGAGGTCCACCGC CCC
OBC200	ACT1_1F	TGGATTCCGGTGATGGTGTT
OBC201	ACT1_1R	TCAAAATGGCGTGAGGTAGAGA
OBC136	TIF4632_1F	CTCCACCATATACACACCAACCA
OBC137	TIF4632_1R	TTCACCAGTTTTCGTAGTGATTTCTATT
OBC132	TIF4631_1F	AAACTGCTCACCCGACACAAT
OBC133	TIF4631_1R	AGCCACGTTGTTGCTGAGATT

2.2.2 Plasmid construction

The entire *TIF4631* open reading frame (START to STOP) was cloned into the multi-cloning site of the p416TEF vector (pBC01; ATCC#: 87368) as a BamHI-EcoRI fragment

amplified from yeast genomic DNA (Strain S288C, Invitrogen) using standard methods with the following primers: Forward(5')=GCAGGGATCCatgacagacgaaactgctcacccgacacaa; Reverse(3')=GCAGGAATTCttactc ttcgtcatcactttctcccattaatgcactgaac (*TIF4631* sequence is in lowercase). Integrity of the construct was confirmed by sequencing. This generates a low copy (CEN6/ARSH4) vector with *TIF4631* under control of the constitutively active, moderately strong translation elongation factor 1-alpha promoter (nulceotides -402 to -1 of the *TEF2* gene) and *CYC1* terminator [171]. This is pBC02.

2.2.3 Plasmid rescue

Wild type (YBC87) and *tif4631*Δ (YBC88) yeast strains were transformed with empty (p416tef/pBC01) or *TIF4631* expressing (pBC02) plasmids using the standard lithium acetate/TRAFO method [169] and positive transformants were selected for on synthetic complete dextrose (SCD) -URA. Plasmid containing colonies were then grown to mid log phase in SCD(-URA). Immediately prior to the addition of cycloheximide (0.1mg/ml), 25mls of culture was removed and used to determine Tif4631 levels as described in the *Western Blot Analysis* section of *Materials and Methods*. Polysomes were prepared and analyzed from the remaining culture as described in the *Preparation And Fractionation Of Polysomes* section (2.2.7). Doubling time was determined as described in the *Doubling Time Determination* section (2.2.6) except cultures were grown in SCD(-URA) instead of YPAD to ensure plasmid maintenance.

2.2.4 eIF4G ortholog detection

To identify eIF4G orthologs we first performed literature searches [157-159, 172-178]. Next, we utilized the Ensembl Genome Browser (www.ensembl.org, [179]) to identify metazoan orthologs of the yeast isoforms (*TIF4631* and *TIF4632*). Finally, to identify additional orthologs throughout eukaryota, the MIF4G domain from *TIF4631* (amino acids 607-850 in the ORF translation from Saccharomyces Genome Database (SGD); www.yeastgenome.org) was input into the PhyloBuilder Web Server (phylogenomics.berkeley.edu/phylobuilder/, [180]). A Global-local search was performed with 10 SHMM and 3 PSIBLAST iterations; all other parameters were left as defaults. The output can be found here: `_`. Neighbor joining phyologenetic tree construction allowed us to distinguish a class of nuclear localized MIF4G domain containing proteins with no assigned translational function, exemplified by *SGD1* (YLR336C) in yeast, from putative eIF4G orthologs. The number of eIF4G encoding genes was then tallied for different organisms and the results were mapped to a eukaryotic phylogenetic tree based on genome sequence (www.itol.embl.de; version 1.6.1; [181]; Figure 2.1).

2.2.5 *Saccharomyces cerevisiae* eIF4G isoform sequence alignment

MUSCLE (www.drive5.com/muscle/, v3.7, [182]) was used to align the full length coding sequences of *TIF4631* and *TIF4632* (obtained from the Saccharomyces Genome Database; http://yeastgenome.org). BLOSUM62 scores indicative of alignment quality were averaged over a sliding window of 3 amino acids (i.e. the score at each position represents the average of it and each residue on either side). The averaged values were then used to

create a false colored heat map. Code to calculate averages and prepare heat map is available in Appendix I.

2.2.6 Doubling time determination

Five-milliliter yeast cultures were grown from a single colony to saturation and used to inoculate 25mLs of YPAD at 0.05 OD_{600}. The OD_{600} of the culture was monitored approximately hourly until readings began to plateau. The logarithmic portion of the growth was estimated from log plots, an exponential curve was fit to this portion ($y=Ae^{kt}$; where y is the culture density at any given time, t, and A and k are constants given when the data is fit to this equation), and the doubling time (DT) calculated (DT=LN(2)/k).

2.2.7 Preparation and fractionation of polysomes

Polysomes were prepared as described previously [183, 184]. Specifically, yeast cultures were grown to an OD_{600} of 1.0 at 30°C in YPAD and transferred into prechilled centrifuge bottles containing $1/100^{th}$ the culture volume of freshly prepared 10mg/ml cycloheximide in water (final concentration 0.1mg/mL cyloheximide in culture). Cells were harvested (~8Kxg), washed once in 50mLs of cold lysis buffer (20 mM HEPES-KOH, pH 7.4, 2 mM Magnesium Acetate, 100 mM Potassium Acetate, 0.1 mg/ml *fresh* cycloheximide, 3 mM *fresh* dithiothreitol (DTT)), harvested as before, resuspended in 20mLs of cold lysis buffer, transferred to 50ml centrifuge tubes, and harvested at ~5Kxg. Pellets were resuspended in 1.5 mL lysis buffer plus 0.2 U/µl RNasin (Promega) per gram of cell pellet, and then 5 g of cold 0.5mm glass beads per gram of cell pellet was added. Cells were lysed by vortexing for 3 minutes (six 30 second intervals with at least 1 minute on ice between). The crude extracts were spun at 10,000 rpm in a microcentrifuge at 4°C for 20 min, and the resulting supernatant was applied to the gradient immediately (see below) or flash-frozen in liquid nitrogen and stored at -80°C for subsequent analysis.

For polysome analysis/fractionation, 10 OD_{260} units of polysomal extract were applied to an 11-ml 10 to 50% sucrose gradient (prepared in 1X lysis buffer). When measuring the concentration of the lysate, an OD_{260} reading of 100 is equivalent to 100 OD/mL and therefore 0.1mLs would be loaded onto the gradient; lysates were typically 100-200 OD/mL. The gradients were spun in a Beckman ultracentrifuge using the SW41 rotor at a speed of 40,000 rpm for 1 h 45 min. The resulting samples were "pushed" out of the centrifugation tube from the bottom using Fluorinert FC-40 (Sigma F9755) at 1.5mL/min and visualized using a UA6 UV/Vis detector (ISCO).
For polysome quantification, the trace was scanned, baselines were added (for the Monosome peak a straight line connecting the right base of the 40S peak with the right base of the 80S peak and a vertical line at the junction of the 60S and 80S peak defined the boundaries; for the Polysome peak a straight line connecting the right of the 80S peak to the baseline following the last polysome was used) and areas were calculated based on the number of pixels in the area sectioned off by the baselines and polysome trace.

2.2.7 Microarray analysis

2.2.7.1 Cell growth, lysis, and RNA isolation

For polysomal RNA samples, cell growth, lysis, and polysomal RNA separation were performed as described above (see *Preparation and fractionation of polysomes*), except 50OD units were loaded onto a 35mL 10-50% sucrose gradient and spun at 27,000 RPM for 3.5hrs in a Beckmann SW-28 swinging bucket rotor. Polysomal RNA containing portions were isolated (n+2 peak on, see Figure 2.9A) and proteins were removed by adding 2.25 volumes of 8M guanidine hydrochloride and vortexing. To precipitate the RNA, 3.25 volumes of isopropanol was added and the mixture was incubated overnight at -20°C. The precipitated RNA was harvested at 30,000 x *g* for 30min, resuspended in 10mM Tris pH8, 1mM EDTA (TE buffer), and precipitated again by adding 0.1 volumes of 3M Sodium Acetate pH 5.3 and 2.5 volumes of 100% ethanol. RNA was either stored precipitated or harvested as above, resuspended in water, and quantified with a UV spectrophotometer.

2.2.7.2 cDNA synthesis and labeling, array fabrication, and hybridization

10-20µg of RNA (same amount for every strain within a single experiment but different in different experiments) was pre-annealed to a 1:1 mixture of oligo dT and random primers (0.32µg/µL each) by incubation at 70°C for 10min followed by 10min on ice. cDNA was synthesized at 42°C overnight using aa-dUTP containing nucleotides (1mM A/G/CTP, 0.4mM TTP, 0.6mM aa-dUTP) and Roche Transcriptor (0.75µL/rxn). Remaining RNA was acid hydrolysed and cDNA purified using Qiagen MinElute columns. The resulting cDNA was labeled with Cy3 or Cy5 (GE healthcare PA23001 or PA25001) via the aa-dU nucleoside, and hybridized, using the MAUI hybridization system (BioMicro), to custom cDNA microarrays containing probes for every Saccharomyces Genome Database (SGD)-annotated open reading frame as well as several non-coding regions. A detailed protocol for array printing, cDNA synthesis, array processing and hybridization, along with details about the oligonucleotide probes utilized (Operon AROS and YBOX sets) is available at the University of California, San Francisco, Center for Advanced Technology (UCSF CAT) website: http://cat.ucsf.edu.

2.2.7.3 Experimental design

To determine relative polysome abundance, cDNA was synthesized from an equivalent amount of polysomal RNA from each strain (wild type, *TIF4631homo*, *TIF4632homo*). Wild type polysomal RNA from that specific biological replicate was used as a common reference and cDNA from each of the three strains was compared to it directly on a single two-color array.

2.2.7.4 Statistical analysis

Two biological replicates of each strain were analyzed in all experiments. Dye-swap technical replicate pairs were included for each of the biological replicates. Standard pre-processing (manual removal of low quality/physically damaged spots, background correction (normexp method)) and normalization (print-tip loess, intensity based quantile, and ratio based scale) methods were applied [185, 186]. A pooled correlation method was applied

to within-array replicate spots, a linear model was fit to the data, and empirical Bayes-based statistical analysis of differential expression was performed (employing Benjamini & Hochberg multiple testing correction) using the limma package from Bioconductor [187-189]. Detailed analysis code is in Appendix I.

2.2.7.5 Raw data

All data is MIAME compliant and the full set of raw and processed data, along with detailed methods regarding data processing have been deposited in the ArrayExpress (http://www.ebi.ac.uk/arrayexpress/) database under accession number E-MEXP-2399.

2.2.8 Polyclonal antibody production

Peptides corresponding to amino acids 4-24 of Tif4631 (ETAHPTQSASKQESAALKQTG), 4-24 of Tif4632 (QRGPPPPHPQQANGYKKFPPH), and 565-582/525-542 of Tif4631/Tif4632 (GPKEEV/IAPLVPSANRWV/IPK) were synthesized at a campus core facility (with terminal cysteines added), conjugated to a carrier protein using Pierce's Imject Maleimide-Activated mcKLH Kit (#77611) (via terminal cysteines), and used to immunize rabbits. No affinity purification was performed and the specificity of the antibodies was confirmed in western blots utilizing rabbit serum (see *Western Blot Analysis* section) on strains missing or over-expressing each eIF4G isoform (Figure 2.4, 2.5, 2.7, and data not shown).

2.2.9 Immunoblot analysis

20mLs of culture was removed from polysome culture immediately prior to addition of cyloheximide (OD_{600} ~1.0) and harvested at ~4Kxg. The cell pellet was washed with 5mls of Harvest Buffer (20mM Hepes pH 7.4, 2mM MgOAc, 100mM KOAc), pelleted again as before, resuspend in 1mL of harvest buffer, transferred to an eppendorf, pelleted with brief (10sec) spin, supernatant was aspirated off, and the pellet was lysed immediately (see below) or stored pellet at -20°C.

For lysis, the pellets were resuspended in 200µL of lysis buffer (20mM Hepes 7.4, 2mM EDTA, 0.5% Triton X-100, 4mM DTT, 1mm PMSF, 1x Roche Protease Inhibitor (1 tablet/10mL)) and 200µL of 0.5mm glass beads were added. The cells were lysed by vortexing (bead beating) 6x30sec with 30sec on ice between. The lysate was clarified by spinning at 16k x *g* in a microfuge for 20min at 4°C. 100µL of the supernatant was removed and A280 was measured. All lysates were adjusted to same concentration (~30mg/ml) with lysis buffer and 4x SDS sample buffer (200mM Tris-Cl pH6.8, 40% glycerol, 8% SDS, 0.04mg/mL bromophenol blue, 20% β-mercaptoethanol) was added to 1x. These lysates were diluted 1:10 (this is the "1X" input) and then serially diluted in 1X SDS sample buffer to make 0.5X and 0.25X inputs.

Samples were separated by SDS-polyacrylamide gel electrophoresis (SDS-PAGE) and transferred to a polyvinylidene difluoride membrane (Immobilon-P (Millipore IPVH07850)) at 110V for 1 hour and 25 minutes, blocked in 4% dry milk in Tris-buffered saline with Tween (TBS-T; 20 mM Tris-HCl, pH 7.4, 500 mM NaCl, 0.05% Tween 20) for

30 min, incubated with rabbit serum containing the desired antibody (see *Polyclonal Antibody Production* section 2.2.8 for details) at 1:2000 in TBS-T for 1hr at room temperature, washed 4x5min with TBS-T, incubated for 1hour at room temperature with a goat anti-rabbit horseradish peroxidase-conjugated secondary antibody at a 1:5000 dilution in TBS-T was used for the primary hybridization, washed 4x10minutes with TBS-T, and detected with Amersham chemiluminescent reagents and Kodak Film.

Western blots were quantified using ImageJ [190]. Three exposures were taken for each blot and the exposure displaying the most linear relationship of the dilution series was chosen for analysis (this may have been different for the eIF4G and cross reacting band). To control for loading variability, the eIF4G band was first normalized to a lower molecular weight, cross-reacting band, which was not an eIF4G breakdown product as it was present in the eIF4G deletion strain. To compare between strains, the intensity of normalized eIF4G bands for a particular dilution were compared (e.g. the normalized "1X" eIF4G band in the deletion strain was compared to the normalized "1X" eIF4G band in the wild type strain), the ratios from different dilutions subsequently averaged, and the mean, and associated error, of several experiments calculated. To compare total levels of eIF4G between strains, an estimate of the relative quantity of isoforms in wild type yeast cells was used and the following formula employed:

$$eIF4G_{strain\,1}/eIF4G_{wild\ type} =$$
$$(eIF4G1/\ (eIF4G1+eIF4G2))*(eIF4G1_{strain\,1}/eIF4G_{wild\ type}) +$$
$$(eIF4G2/\ (eIF4G1+eIF4G2))*(eIF4G2_{strain\,1}/eIF4G_{wild\ type})$$

eIF4G1 levels were estimated to be 9000 molecules/cell and eIF4G2 to be 3000 molecules/cell in yeast ([191]). Therefore the formula simplifies to:

$$eIF4G_{strain\,1}/eIF4G_{wild\ type} =$$
$$(3/4)*(eIF4G1_{strain\,1}/eIF4G_{wild\ type}) +$$
$$(1/4)*(eIF4G2_{strain\,1}/eIF4G_{wild\ type})$$

2.2.10 Reverse-transcription quantitative PCR analysis (RT-qPCR)

To measure total relative transcript abundance, a 7.5mL yeast culture was grown to mid-log phase (OD_{600} ~= 1.0) in rich media and cells were harvested at 4000g for 5 minutes. Cell pellet was resuspended in 0.5mL of phenol (pH 4.1) by vortexing and 0.5mL of AES buffer (50mM Sodium Acetate pH 5.2, 10mM EDTA, 1% SDS) was added. The samples were heated at 65°C for 10 minutes, vortexing every minute, then cooled on ice for 5 minutes and spun at max speed in a microfuge for 5 minutes. The aqueous layer (~0.5mL) was transferred to a fresh tube and 50µL of 3M sodium acetate pH 5.3 and 550µL of isopropanol were added to precipitate the RNA. The precipitate was harvested at max speed in a microfuge for 30 minutes at 4°C, and the pellet washed with 1mL of 70% ethanol, then dried, and resuspended in water. RNA quality (based on rRNA bands) was inspected on an agarose gel. The sample was treated with 1U of RQ1 DNAse (Promega) for every microgram of RNA (as judged by A_{260}) at 37°C for 30 minutes to remove any contaminating DNA.

For determining relative polysome abundance, polysomal RNA was prepared as described above (see *Preparation and fractionation of polysomes* (and *Microarray* sections). 5µg of RNA was pre-annealed to a 1:1 mixture of oligo dT and random primers (0.32µg/µL each) by incubation at 70°C for 10min followed by 10min on ice. cDNA was synthesized at 42°C overnight using and Roche Transcriptor (0.75µL/rxn) and deoxynucleoside triphosphates (1mM each A, T, C, and G). Remaining RNA was acid hydrolysed (100mM EDTA, 200mM NaOH, 65°C 10min, neutralize with 0.5M HEPES pH7) and cDNA purified using Qiagen MinElute columns. 1-1.5ng of cDNA was used in qPCR reactions along with 400nM each primer and 1x Fast SYBR Green Master Mix from Applied Biosystems (*TIF4631*-F: AAACTGCTCACCCGACACAAT; *TIF4631*-R: AGCCACGTTGTTGCTGAGATT; *TIF4632*-F: CTCCACCATATACACACCAACCA; *TIF4632*-R: TTCACCAGTTTTCGTAGTGATTTCTATT; *ACT1*-F: TGGATTCCGGTGATGGTGTT; *ACT1*-R: TCAAAATGGCGTGAGGTAGAGA). Following a 1minute, 95°C enzyme activation step 40 cycles of 95°C, 10 seconds and 60°C, 45 seconds were performed and a melting curve analysis followed. Samples were measured in triplicate and a no reverse transcriptase control was included to detect incomplete genomic degradation. Relative amounts of each transcript were calculated from a standard curve and cDNA samples normalized using actin (*ACT1*).

2.3 Results

2.3.1 Diverse eukaryotes express multiple eIF4G isoforms

Multiple isoforms of eIF4G were first identified in plants and have been characterized in yeast, fly, worm and mammalian cells [170]. To analyze the extent of multiple eIF4G isoform conservation in light of previous studies [170], we performed phylogenetic-based homology detection using the MIF4G domain from *TIF4631* (amino acid residues 607-850; PFAM: PF02854) ([170]; see Section 2.2.4 for details). Multiple MIF4G-containing proteins with homology to characterized eIF4G family members were found in the genomes of a wide variety of eukaryotes, from protists (trypanosomes, plasmodia) to fungi (yeast) to plants (wheat, *Arabadopsis*) to animals (worms, flies, human) (Figure 2.1). A more detailed, interactive phylogenetic tree containing the complete results of this analysis is available at: http://phylogenomics.berkeley.edu/book/book_info.php?book=bpg081634d. Organisms where we identified zero or a single putative eIF4G-encoding locus may contain additional highly divergent loci missed by our analysis or may encode multiple isoforms from a single locus, as is the case in *C. elegans* [170]. These analyses highlight the diversity of eukaryotes expressing multiple eIF4G isoforms and suggest specialized function of eIF4G variants may be a conserved mechanism of translational control.

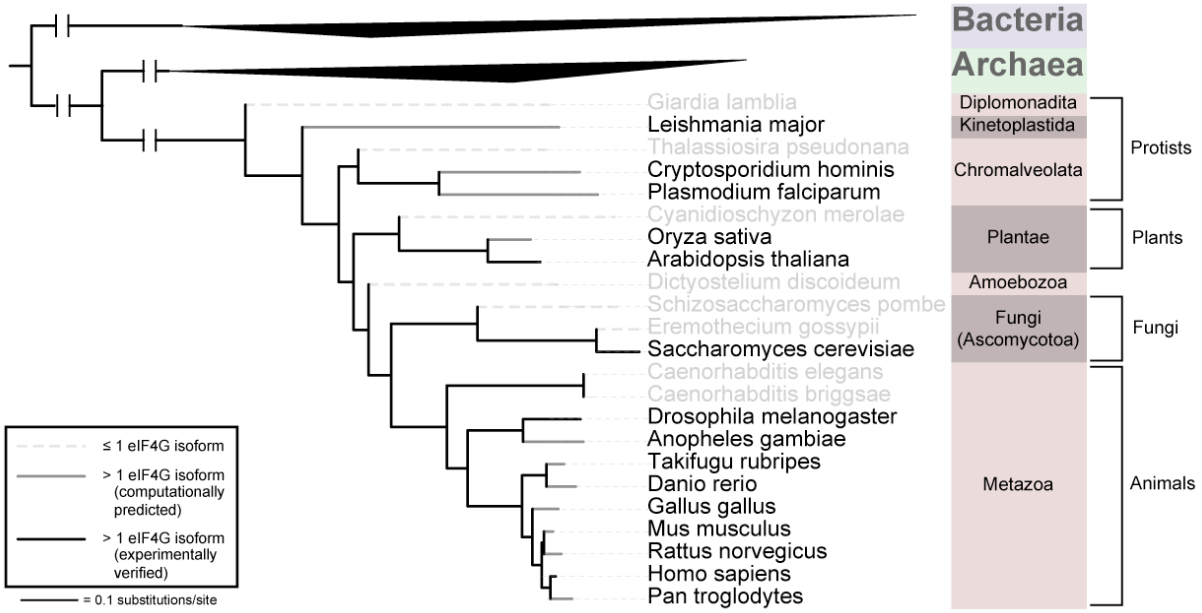


Figure 2.1. Eukaryotic eIF4G isoform distribution. The number of eIF4G-encoding loci in the genomes of diverse eukaryotes was determined through a combination of literature searches and computational homology detection (See Results and Section 2.2.4 for details). A portion of the results were then mapped to a phylogenetic tree constructed based on the genome sequence of broadly representative eukaryotes [181]. Organisms that encode multiple eIF4G isoforms are indicated by solid branches and black text, while those encoding one or fewer have dashed branches and gray text. Black lines represent experimentally verified findings while gray lines indicate computationally predicted orthologs.

Saccharomyces cerevisiae expresses two forms of eIF4G that are 51% identical (72% similar), encoded by *TIF4631* (eIF4G1) and *TIF4632* (eIF4G2). Both isoforms share an evolutionarily conserved domain architecture, with the PABP- and eIF4E-interacting portions in the N-terminus and the region responsible for eIF4A interaction and ribosome connectivity in the C-terminus (Figure 2.2A). Despite a high degree of overall homology, the sequence similarity of the yeast eIF4G isoforms is heterogeneously distributed and heavily enriched in the C-terminus. Closer inspection of a global pairwise alignment reveals that sequences surrounding and within some of the functionally conserved N-terminal domains show minimal sequence homology (Figure 2.2B). Thus, unique biochemical properties could contribute to the distinct translational capacities of eIF4G variants and function in translational control.

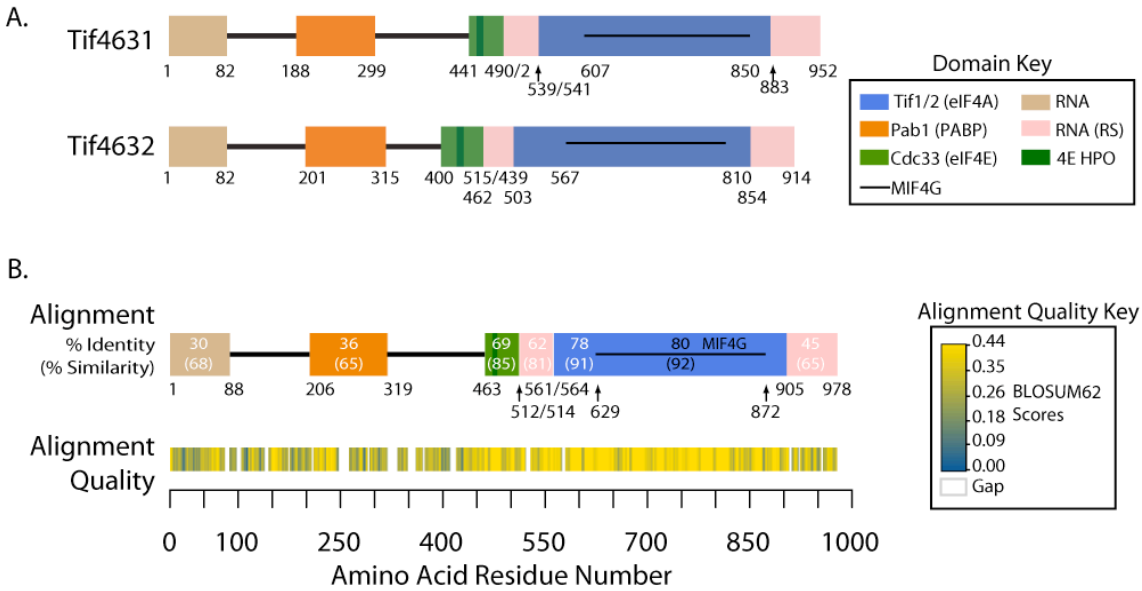


Figure 2.2. eIF4G isoforms from *Saccharomyces cerevisiae*. (**A**) Schematic of the two eIF4G isoforms from *S. cerevisiae* (encoded by *TIF4631* and *TIF4632*). Experimentally verified (except RNA binding segments of Tif4632, which are computationally predicted) domains of interaction are shown as colored blocks with binding partners indicated on the right and boundaries (amino acid positions) below [59, 156, 192-195]. RNA (RS) = Arginine-Serine rich RNA binding domain; 4E HPO = Y(X)$_4$LΦ motif of eIF4E (Φ = hydrophobic amino acid). (**B**) A pairwise alignment of *TIF4631* and *TIF4632* protein sequences was generated using MUSCLE [182] and a schematic of the consensus is shown. Domain homologies are indicated as a percentage of amino acid identity (similarity). Alignment quality (based on BLOSUM62 Scores; key on right) is indicated in the heat map below, where yellow is high quality, blue is low quality, and white is a gap (where sequence in the consensus originates from a single isoform). The score at each residue represents a sliding window average of 3 amino acids (See section 2.2.5 for details).

2.3.2 Deletion of *TIF4631* causes a translation initiation defect

Prior analysis of eIF4G variants in yeast revealed that deletion of *TIF4631* (*tif4631*Δ) impairs cell growth, whereas the *tif4632*Δ strain grows at wild type rates under standard laboratory conditions [173, 196]. To investigate whether this decreased growth rate relates

to a protein synthesis defect, we examined the translational profile of wild type (YBC87), *tif4631*Δ (YBC88) and *tif4632*Δ (YBC89) cells. Yeast grown at 30°C in rich media to mid-log phase (OD_{600} = 1.0) were harvested and lysed in the presence of cycloheximide to inhibit ribosome run-off [26, 27, 197-199]. Lysates were fractionated on sucrose density gradients to resolve free ribosomal subunits (40S and 60S), intact 80S ribosomes (monosomes or "M") and polysomes ("P") (Figure 2.3A).

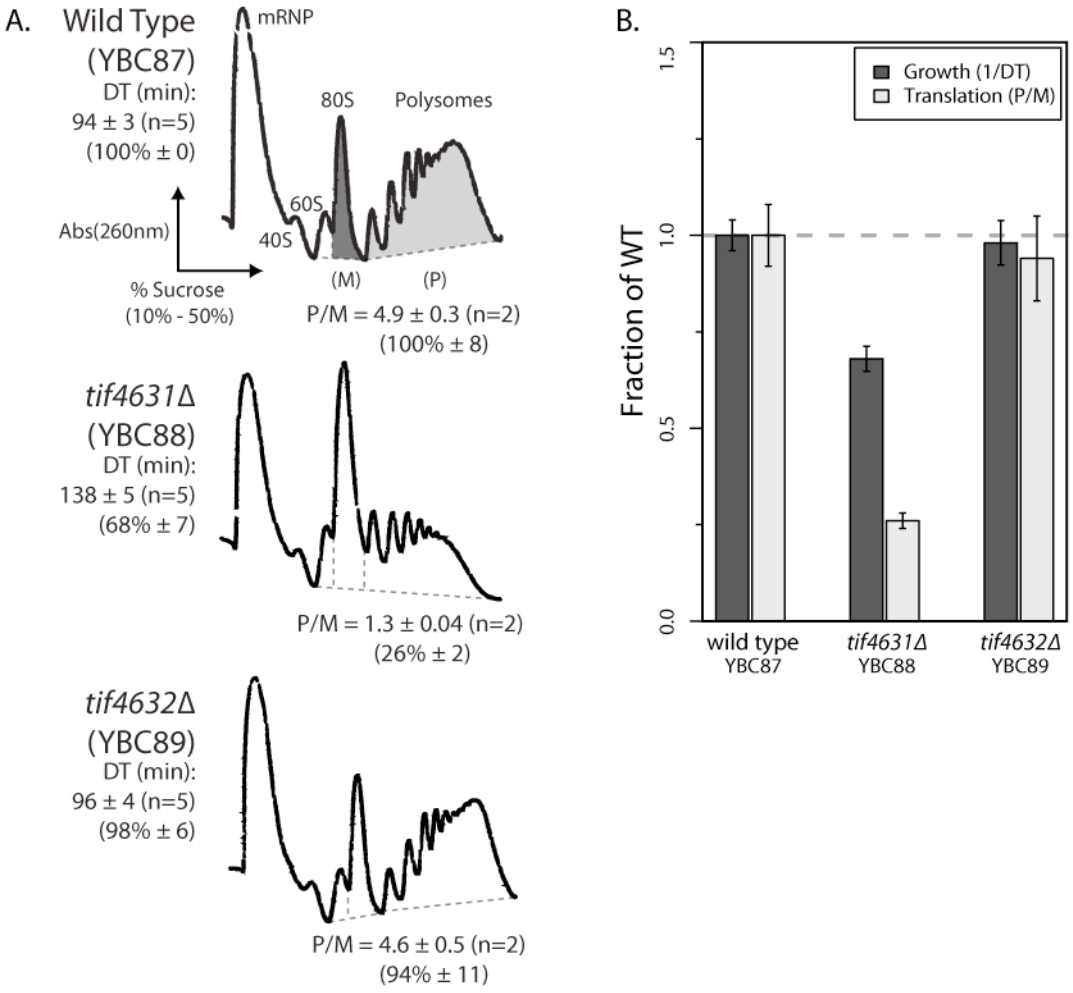


Figure 2.3. Polysome profile analysis of eIF4G deletion strains. (**A**) Polysome profile analysis of wild type (top), *tif4631*Δ (middle), *tif4632*Δ (bottom) strains. Mid log-phase cultures (OD_{600} = 1.0) were lysed and a normalized (by A_{260}) amount of lysate was separated on a 10-50% sucrose gradient (See section 2.2.7). The displayed trace represents absorbance at 260nm (vertical axis) throughout the gradient from top (left) to bottom (right). Peaks corresponding to free ribosomal subunits (40S, 60S), intact ribosomes (80S) and polysomes are indicated. The area underneath the monosome (80S, M; dark gray) and polysome (P, light gray) peaks were determined for multiple biological replicates (n = 2) and the mean polysome:monosome (P/M) ratio as well as the mean doubling time (DT; n = 2) are provided below and to the left of each trace, respectively. Normalized (percentage of the wild type) translation (P/M) and growth (1/DT) values are indicated in (parentheses) and summarized in bar chart form in (**B**).

*tif4631*Δ cells exhibit a clear translation initiation defect, as shown by a decrease in the number of polysomes with a concomitant increase in the number of monosomes (Figure 2.3A YBC88 profile versus YBC87), whereas *tif4632*Δ cells have a polysome profile qualitatively indistinguishable from that of the wild type strain (Figure 2.3A YBC89 versus YBC87). Quantification of the areas under the monosome and polysome peaks shows that the P/M ratio of *tif4631*Δ cells is 26% ± 2% (1.3/4.9) of wild type, while the *tif4632*Δ cells P/M ratio is statistically indistinguishable from wild type (4.6 ± 0.5 versus 4.9 ± 0.3) (Figure 2.3B). Previously, deletion of *TIF4631* was observed to decrease the levels of free 60S subunits and have no effect on polysome content [173]. In our studies, no differences in 60S levels were observed and the polysome defect was reproducible. Both the growth and translation initiation phenotypes could be rescued by exogenous expression of *TIF4631* (Figure 2.4), suggesting they arise specifically from the absence of *TIF4631* and not another mutation in the strain. Discrepancies between previous studies and our current findings could result from a difference in strain background (see below). Therefore, deletion of the gene encoding eIF4G1 (*TIF4631*), but not eIF4G2 (*TIF4632*), results in translation initiation defects.

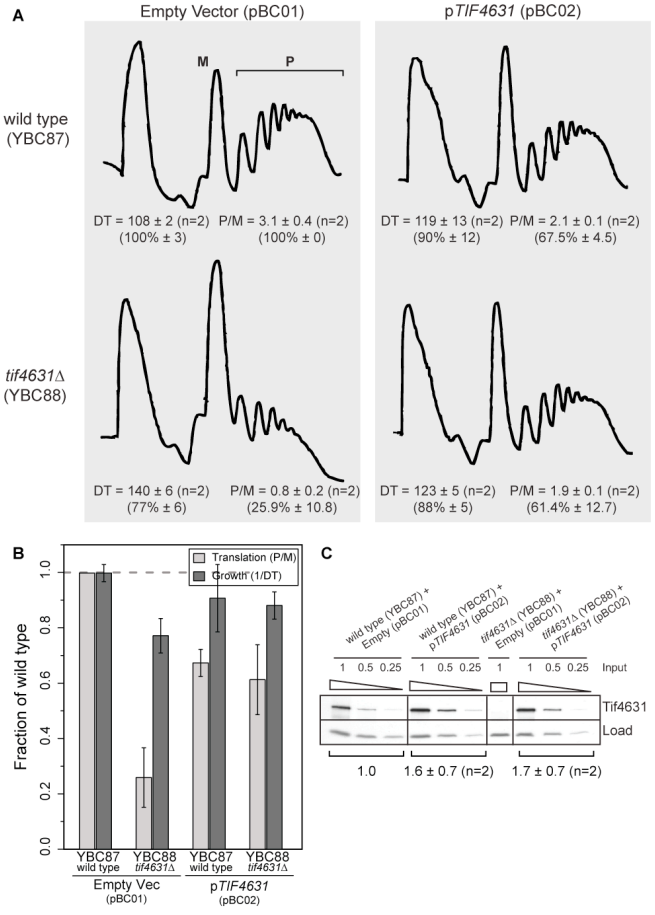


Figure 2.4. Growth and polysome profile analysis of strains expressing exogenous *TIF4631*. (**A**). Polysome profile analysis of wild type (YBC87; top row) and *tif4631*Δ (YBC88; bottom row) cells containing either an empty vector (left column) or one expressing *TIF4631* from the Translation Elongation Factor (TEF) 1-alpha constitutive promoter (p*TIF4631* (pBC02); right column) as in Figure 2.3. Peaks corresponding to monosomes (M) and polysomes (P) are indicated. Mean polysome:monosome ratios (P/M; n = 2) and 1/doubling times (DT; n = 2) are provided below each trace with fraction of the wild type values indicated in parenthesis. (**B**) Bar chart summarizing the normalized (fraction of the wild type strain carrying the empty vector) translation (P/M) and growth rates (1/DT) of strains in (A). (**C**) Tif4631 protein levels for the strains in (A). Two-fold serial dilutions of indicated plasmid-strain combinations (see labels above) were probed with a Tif4631-specific antibody and Tif4631 band intensity was normalized by a cross reacting species (Load). The normalized bands from identical input amounts were compared, and the average (n=2) amount of Tif4631 as compared to the wild type stain with an empty vector is indicated below each of the strain-plasmid combinations.

2.3.3 *tif4631*Δ-associated phenotypes correlate with overall eIF4G levels

When performing the polysome profiling experiments, we noted that *tif4631*Δ strains from two different backgrounds exhibited phenotypes to differing degrees. Specifically, YBC90 (YBC86 *tif4631::kan*) shows more pronounced growth and translation defects than does YBC88 (YBC87 *tif4631::kan*) (Figure 2.5A versus Figure 2.3A). Since eIF4G is thought to be the limiting translation factor in yeast [191, 200] and phenotypes only result from deletion of the gene encoding the more highly expressed isoform [191, 201, 202], we hypothesized that the growth and translation defects might be related to a reduction in the overall levels of eIF4G (i.e. the sum of Tif4631 and Tif4632) rather than the absence of an isoform-specific functionality. To examine this further we determined the protein levels of both eIF4G isoforms in wild type, *tif4631*Δ and *tif4632*Δ strains from both backgrounds (Figure 2.5B).

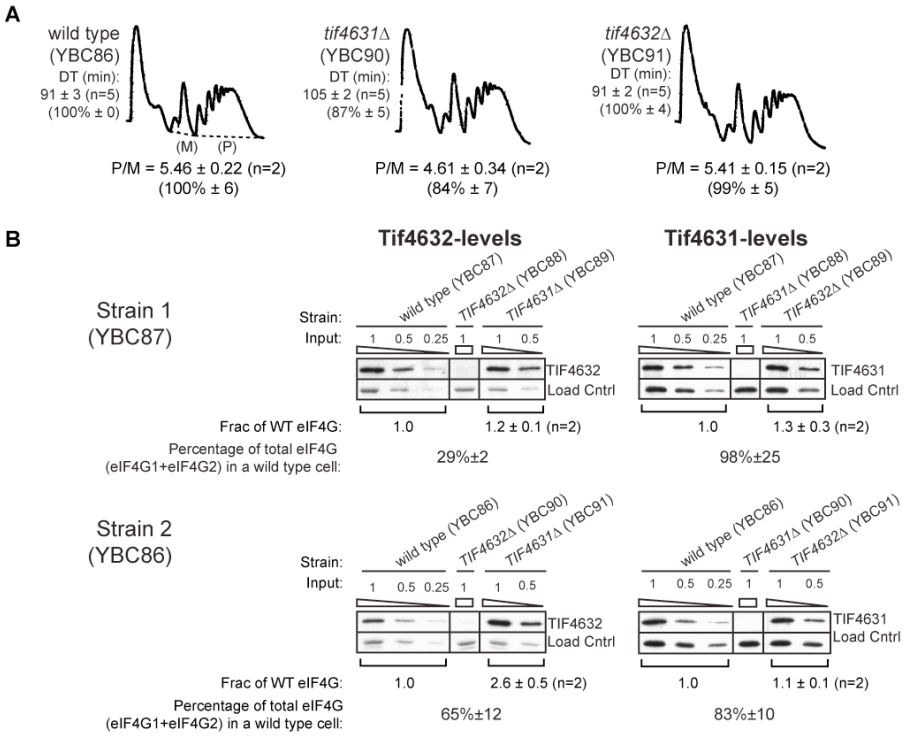


Figure 2.5. Doubling time, polysome profile and eIF4G protein level analysis in another commonly used strain background. (**A**) Growth rate and polysome profile analysis of wild type (left), *tif4631*Δ (middle) and *tif4632*Δ (right) cells from a second commonly used laboratory strain background. Polysomes were prepared from a mid-log (OD_{600}=1) culture of cells grown in rich media as in Figure 2.3. Doubling time (DT) and polysome:monosome (P/M) ratio are provided, with normalized (percentage of the wild type) values indicated in (parentheses) and summarized in (**B**). (**C**) Levels of Tif4632 (left) and Tif4631 (right) were determined in wild type, *tif4631*Δ and *tif4632*Δ strains from both backgrounds (top = YBC87 (from Figure 2.3); bottom = YBC86 (from(A)). Two-fold serial dilutions of lysate from polysome cultures were probed with isoform-specific antibodies, eIF4G band intensities were normalized to a cross reacting species (Load) and compared between wild type and deletion strains. The fraction (frac) of wild type (WT) (determined in two biological replicates) is indicated below each strain. The percentage of *total* eIF4G in a wild type cell, which incorporates genotype and concentration of each isoform, is also provided for the deletion strains (see Section 2.2.9 for details on quantification and calculations).

To accurately determine the levels of specific eIF4G variants, we generated isoform-specific, polyclonal antibodies using peptides corresponding to a unique sequence from the N-terminus of each isoform (see Section 2.2.8). Serial dilutions of lysate were probed and normalized using an unrelated cross-reacting band. The *tif4631*Δ strain that exhibits severe growth and polysome defects (YBC88; Figure 2.3A) maintains Tif4632 at levels similar to the wild type parent, resulting in a significant (~70%) decrease in overall eIF4G as compared to the concentration in a wild type cell (Figure 2.5B top). In contrast, the *tif4631*Δ strain that exhibits only subtle growth and initiation rate defects (YBC90; Figure 2.5A) increases Tif4632 levels approximately 2.5-fold, resulting in a more modest (~30%) decrease in total levels of eIF4G (Figure 2.5B bottom). Reverse Transcription qPCR (RT-qPCR) experiments revealed that this compensatory up-regulation also occurs at the mRNA level and is not a translational control phenomenon (Figure 2.6). Therefore, the severity of *tif4631*Δ-associated phenotypes correlates with overall levels of eIF4G and can be attenuated by the compensatory upregulation of Tif4632.

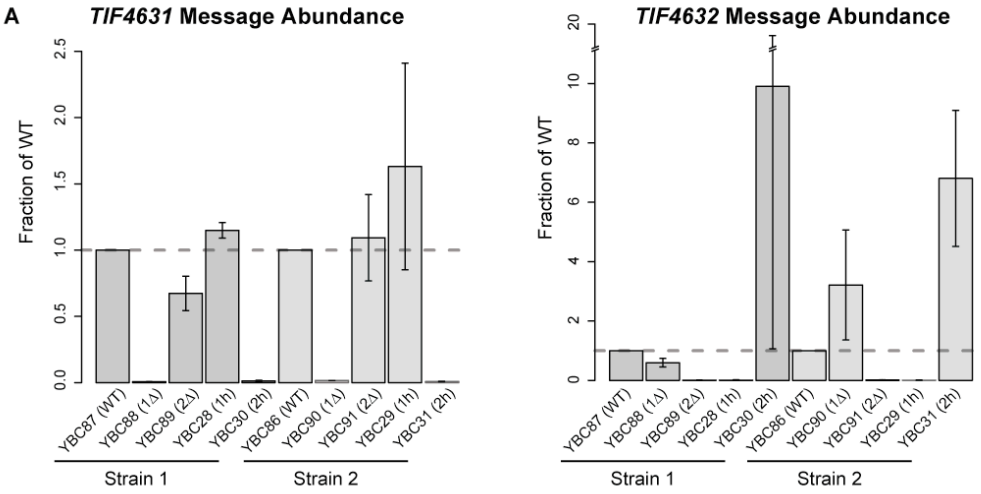


Figure 2.6. eIF4G transcript levels in different yeast strains. (A) *TIF4631* (left) and *TIF4632* (right) transcript levels in mid log phase cultures of wild type, deletion, and homogenic strains were determined by RT-qPCR (see section 2.2.10 for details). Values relative to those in a wild type cell are plotted. 1Δ = *tif4631*Δ, 2Δ = *tif4632*Δ, 1h = *TIF4631homo,* 2h = *TIF4632homo.*

2.3.4 Yeast eIF4G isoforms have a similar capacity to support growth and translation

To rigorously test whether the phenotypes and changes in polysome-associated mRNAs observed in *tif4631*Δ cells result from reduced eIF4G levels or from an eIF4G1-specific function, we created isoform-specific ("homogenic" (*homo)*) yeast strains. The ORF of a single isoform was placed at both eIF4G genomic loci, without disrupting the surrounding non-coding sequence (Fig. 2.7A; Section 2.2.1). Western blotting reveals a modest increase (1.4-fold and 1.2-fold in Strain 1 and 2, respectively) in the level of Tif4631 and a significant (2.8-fold and 4.1-fold) increase in the level of Tif4632 in the homogenic

strains as compared to wild type (Figure 2.7B and 2.7D). Previous work has shown Tif4631 to be expressed at 3 times the level of Tif4632 [191]. Based on this relationship, Tif4632 levels would need to increase to ~4-times that in wild type parent to bring the final eIF4G level to that of a wild type or *TIF4631homo* cell. *TIF4632homo* strain construction achieves this and, therefore, the homogenic strains contain similar overall levels, but distinct forms, of eIF4G.

To determine the capacity of each yeast eIF4G isoform to support general translation, we examined the polysome profiles of the homogenic strains. Both homogenic strains have polysome profiles that are indistinguishable from wild type cells (Figure 2.7C and 2.7E). Therefore, during exponential growth in rich media, both eIF4G isoforms have a similar capacity to support general translation when expressed at levels close to that of the total eIF4G in wild type cells.

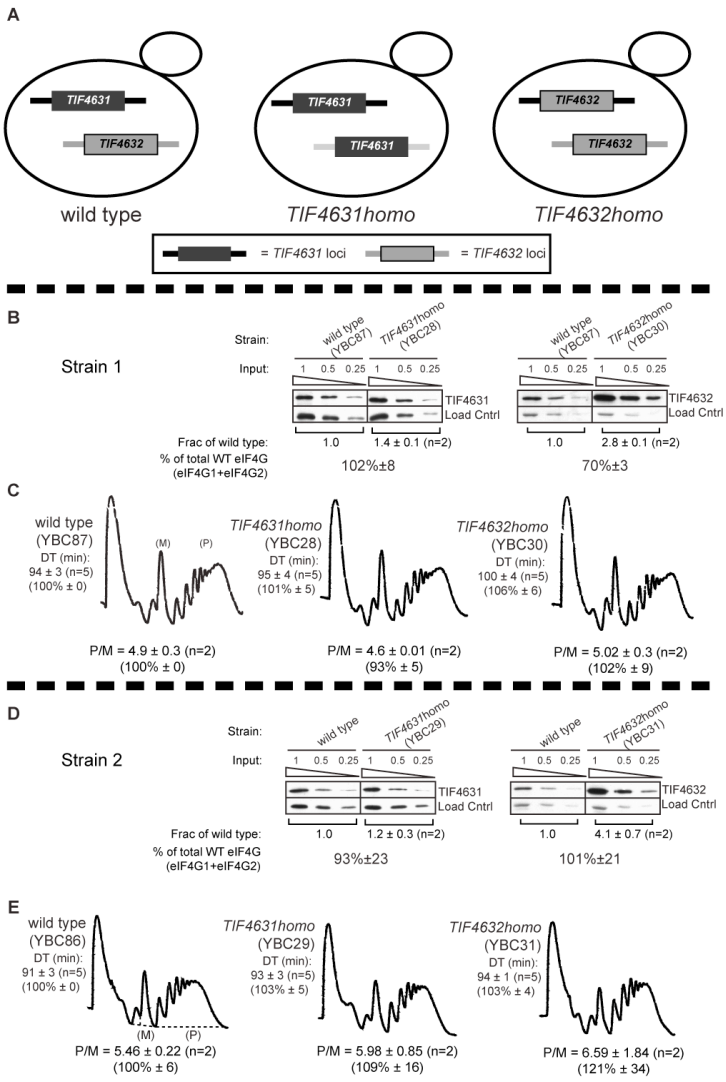


Figure 2.7. Construction, eIF4G protein level, polysome and doubling time analysis of isoform-specific strains. (**A**) Schematic of *TIF4631* (dark) and *TIF4632* (light) loci in wild-type (left), *TIF4631homo* (center) and *TIF4632homo* (right) strains. In each of the *homogenic* strains, the ORF of a particular isoform was placed at both loci without disruption of the surrounding regulatory sequence. eIF4G protein level (**B, D**), polysome profile and doubling time (DT) (**C, E**) analysis of *homogenic* strains in different backgrounds (Strain 1 and Strain 2) as in Figure 2.5.

2.3.5 The global translational profile of eIF4G isoform-specific strains is similar

The results of the previous experiments suggest that eIF4G levels, rather than isoform-specific functions in translation, account for the observed differences in polysome-associated mRNAs in cells lacking the eIF4G1 (Tif4631) isoform. If true, we expected that most or all of these differences would no longer be observed in the homogenic eIF4G yeast strains.

To test this idea, the relative abundance of mRNAs in polysomes isolated from mid log (OD_{600} = 1.0) phase cultures of wild type and homogenic strains grown in rich media were compared using two-color cDNA microarrays (raw data is available at ArrayExpress accession number: E-MEXP-2402). The strong correlation between replicates indicates the high quality of this data (Figure 2.8). The complement of messages in the polysomes of both homogenic strains is similar to that of wild type cells (Figure 2.9B).

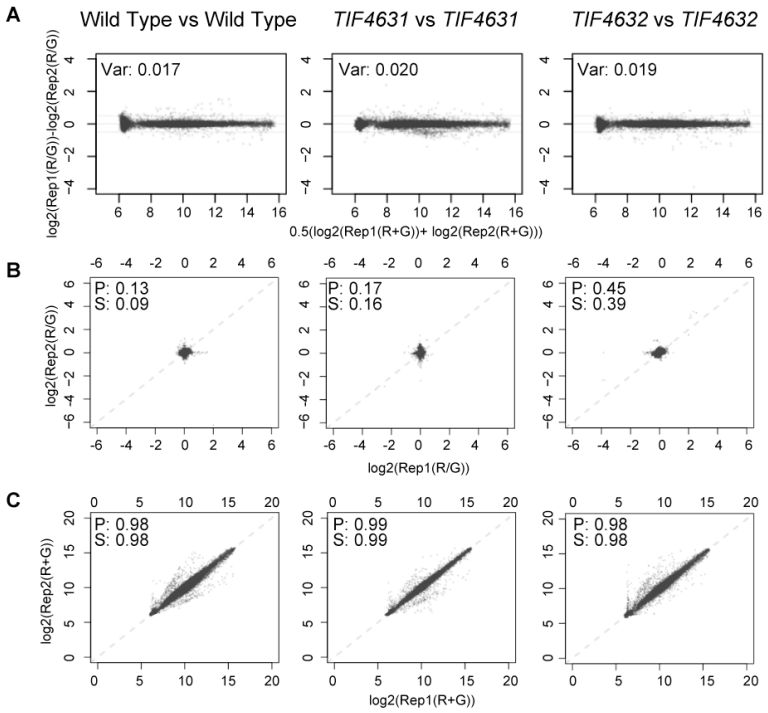


Figure 2.8. Reproducibility of microarray datasets. (**A**). M values (log2(R/G); R = red intensity, G = green intensity) of dye swap technical replicates were sign corrected and averaged for each biological replicate (Rep1 and Rep2) and differences in these values (log2((Rep1a(R/G)-Rep1b(G/R))/2)-log2((Rep2a(R/G)-Rep2b(G/R))/2)) are plotted (vertical axis) against average spot intensity (0.5(log2((Rep1(R+G)+Rep2(R+G)))); horizontal axis). Variances (Var) of these comparisons are indicated on the plot. A tight vertical distribution around zero is indicative of high similarity between biological replicates. M values (**B**) and average spot intensities (**C**) of biological replicates are plotted against each other and the Pearson (P) and Spearman rank (S) correlation of these comparisons is indicated. Low M-value correlations (B) are due to lack of significant differences. High intensity correlation exemplifies high degree of reproducibility.

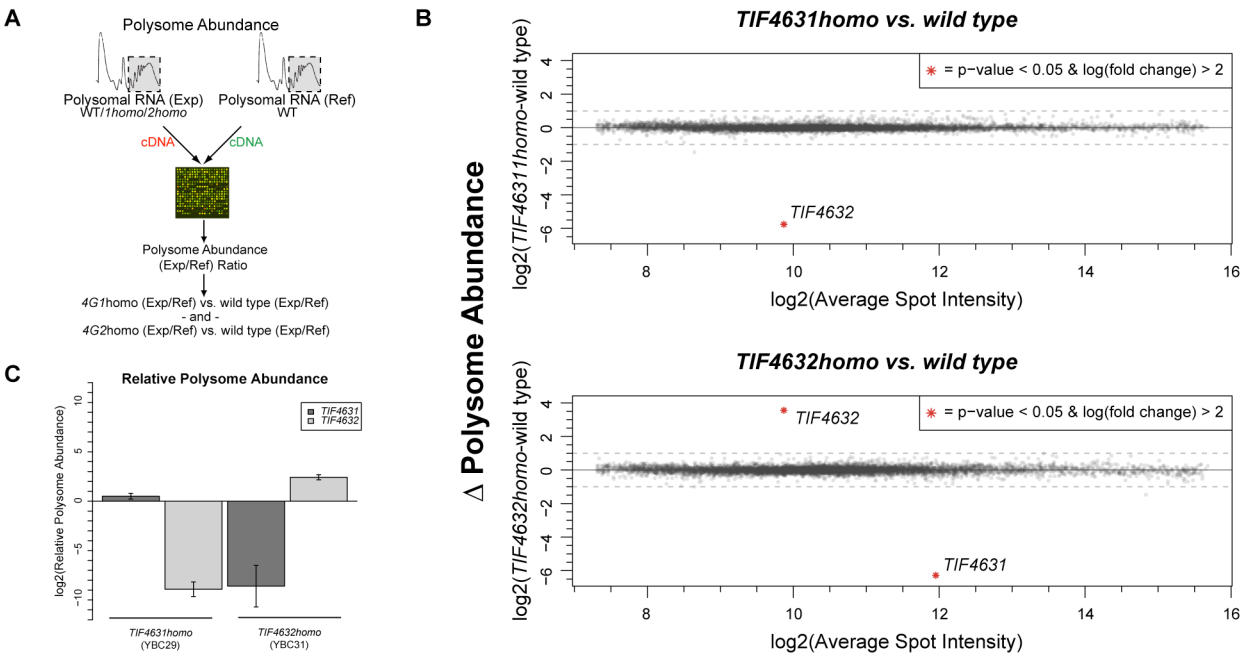


Figure 2.9. Differences in message-specific polysome abundance of wild type and isoform-specific strains. (**A**) Schematic outlining how changes in relative polysome abundance were determined. WT = wild type, *4G1homo* = *TIF4631homo*, *4G2homo* = *TIF4632homo,* Exp = experimental sample, Ref = common reference used in all hybridizations. (**B**) Plots displaying differences in polysome abundance ($\log_2$(*homo*(red/green)) - $\log_2$(wild type(red/green)); vertical axis) versus average spot intensity (0.5($\log_2$(*homo*(red+green)) + $\log_2$(wild type(red+green))); horizontal axis) between *TIF4631homo* (top) or *TIF4632homo* (bottom) and wild type strains. Statistically significant changes in polysome abundance with log fold changes > 2 are indicated by red "*"s. Dashed horizontal lines indicate a 2-fold ($\log_2$(1)) change. (**C**) *TIF4631* and *TIF4632* polysomal transcript levels in mid log phase cultures of each homogenic strain were determined by RT-qPCR (see Section 2.2.10 for details). Values relative to those in a wild type cell are plotted.

The relative polysome occupancy of only one ORF (*YGL049C* (*TIF4632)* was significantly different (p-value <0.05) in *TIF4631homo* cells as compared to wild type (Figure 2.9B). The *TIF4632* gene is missing from the *TIF4631homo* strain and is predicted to exhibit significant differential expression. The non-transcribed spacer region from the rDNA locus, *NTS1-2* , was also mildly upregulated (1.8-fold), although the significance of this is difficult to assess as the overall expression of this feature is low, as predicted based on its location in a transcriptionally silenced region [203].

Only four *bona fide* ORFs exhibit statistically significant polysome perturbation in the *TIF4632homo* strain when compared to wild type. Two (*YHR216W* and *YBR072W*) are less than 2-fold affected (1.8-fold down and 1.7-fold up, respectively), a third, *TIF4631*, which is missing from the *TIF4632homo* strain, is strongly down (77-fold) and finally, *TIF4632*, which was placed at both eIF4G loci, is 3.6-fold upregulated, which corresponds well with the western data presented previously (Figure 2.9B and 2.7E). Reverse Transcription qPCR (RT-qPCR) validated the differential expression of eIF4G genes in polysomal RNA utilized in the microarray experiment as well as additional biological replicate samples from the

homogenic strains (n = 2; Figure 2.9C). Therefore, under the growth conditions tested here, no mRNAs rely strongly on a particular eIF4G isoform for their translation.

2.4 Discussion

As a central player in the initiation process, eIF4G is essential for protein synthesis. The genomes of a wide variety of eukaryotes encode multiple eIF4G variants (Figure 2.1), and evidence in several systems suggests that multiple eIF4G isoforms exist as a means of modifying the profile of translating messages [155, 157-159, 163, 164].

In this study, we attempted to identify messages that rely heavily on a specific eIF4G isoform for their translation, and found none (Figure 2.9). There are several possible explanations for this finding. The preference of mRNAs for a particular eIF4G isoform may not have been sufficient to affect polysome occupancy to a degree that could be detected in our assay. Alternatively, mRNAs displaying strong isoform dependence may not have been expressed under the conditions examined. Notably, we did not observe increased sensitivity of either homogenic strain to salt, osmotic pressure, carbon source, potassium disulfite, rapamycin or wortmannin stress (data not shown). Additionally, isoform-specific strains exhibited similar fitness when competed directly against each other under these various conditions (data not shown). Therefore, both eIF4G isoforms can support expression of proteins essential for fitness under a wide variety of growth conditions.

Although eIF4G isoform-specific effects were not identified, we observed a threshold of total eIF4G protein level (~60-75% of wild type; Figure 2.10), below which translation initiation and growth rates are compromised. Given that eIF4G is thought to be the limiting translation factor in yeast [191, 200], we were surprised to find that cells can tolerate up to a ~30% reduction in overall eIF4G levels with minimal effects on growth and translation initiation rates. This suggests that either other factor(s) are limiting, or there is some tolerance built into the translation initiation system.

Taken together, our findings support the conclusion that the two functionally overlapping eIF4G genes fortify the translational apparatus by ensuring that adequate eIF4G levels are maintained. Consistent with this idea, we observed cross-talk between expression of the two yeast isoforms. In certain strain backgrounds, deletion of *TIF4631* results in increased levels of the *TIF4632* message and protein (Figure 2.5 and 2.6). The mechanism underlying the compensatory upregulation of *TIF4632* is unknown, but this finding suggests that maintenance of adequate eIF4G levels exerts sufficient selective pressure to necessitate the development of such a system. A similar phenomenon has also been observed with different forms of eIF4GI in mammalian cells [204]. The role of eIF4G in mammalian cell cycle control [164], as well as temporal expression of eIF4G isoforms during yeast cell cycle progression [205], underscore the importance of tight control over eIF4G expression.

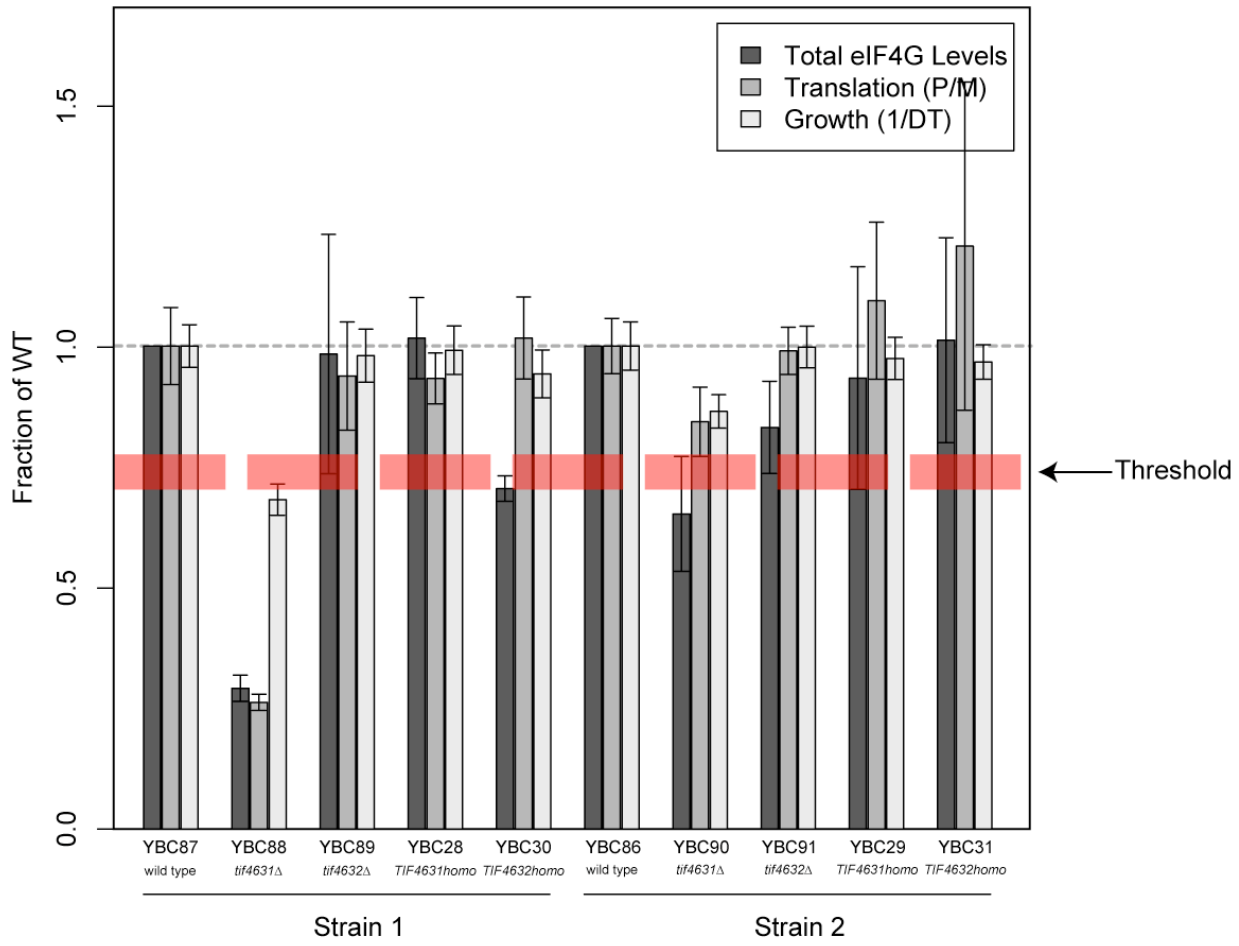


Figure 2.10. Summary of eIF4G level, translation initiation and growth rate data. Summary of total eIF4G levels, polysome:monosome (P/M) ratios and doubling times (1/DT; from Figures 2.3, 2.5 and 2.7) of deletion or homogenic strains normalized to wild type levels of the appropriate strain background (set to 1). The red dashed line indicates a hypothetical threshold of total eIF4G levels, below which growth and translation initiation are compromised.

In this study, we observed phenotypes that correlate with total eIF4G protein levels and are insensitive to isoform type. This distinction is important to consider when interpreting results from experiments where overall amounts of eIF4G, as well as relative isoform abundance, have been modified (as in [163]). The homogenic strains are a powerful tool that distinguishes isoform-specific from level-specific phenotypes. Extensive environmental (e.g. Biolog Phenotype Arrays) and genetic (e.g. synthetic genetic array (SGA)/epistatic miniarray profile (E-MAP)) profiling of homogenic strains has strong potential to identify novel mechanisms of translational control.

Chapter 3

Message-specific effects of limiting eIF4G availability*

* A portion of the work presented in this chapter has previously been published as part of the following paper: Clarkson BK, Gilbert WG, Doudna JA. Functional overlap between eIF4G isoforms in *Saccharomyces cerevisiae*. *PLoS ONE* (2010) 5: e9114.1-e9114.15

Bryan K Clarkson performed the experiments and analyzed the data. Wendy V Gilbert and Jennifer A Doudna assisted with experimental design and manuscript preparation.

3.1 Introduction

Response to environmental cues and maintenance of cellular homeostasis requires regulation at the level of protein synthesis. The majority of known translational control mechanisms target initiation, the rate-limiting step of protein synthesis in which ribosomes are assembled and positioned on a messenger RNA (mRNA). To initiate translation, most eukaryotic mRNAs utilize distal elements (the 5'-7-methyl guanosine (m7G) cap and 3'-poly-adenosine (poly(A)) tail) to interact with a large (> 30) network of polypeptides (reviewed in [21, 22]). Eukaryotic Initiation Factor 4G (eIF4G) is a hub of translation factor interactions. Direct interactions of the cap and poly(A) tail binding proteins (eIF4E and PABP, respectively) with eIF4G circularizes the mRNA and allows regulatory sequences at both ends to influence initiation [80-82]. Notably, a recently identified RNA binding domain in *S. cerevisiae* eIF4G functionally substitutes for PABP binding and bypasses the requirement of message circularization for growth in yeast [85]. eIF4G also directly stimulates the helicase activity of eIF4A and targets it to the cap proximal region of the message via simultaneous interaction with eIF4E [65, 66, 86, 87]. This "activates" the mRNA by creating a conducive binding surface for the small (40S) ribosomal subunit. eIF4G then recruits the 40S to the activated message via contacts (directly in mammals indirectly in yeast) with associated factors and facilitates the "scanning" of this complex to the start codon [57-60, 64]. The translation of certain yeast transcripts requires eIF4G exclusively at stages downstream of ribosome recruitment, suggesting role of eIF4G varies from message to message [90].

Numerous regulatory mechanisms target eIF4G to influence translational efficiency. For example, cysteine proteases, called caspases, cleave eIF4G to regulate translation during apoptosis (reviewed in [9]). Viruses also encode proteases that site-specifically cleave eIF4G, separating the domains required for mRNA (via eIF4E and PABP) and ribosome interaction. These eIF4G fragments simultaneously inhibit host and stimulate translation of viral RNAs (reviewed in [14]). Nutrient limitation reduces eIF4G levels in yeast and mammals [163, 206]. The roles of eIF4G in translation dictate the effects of these different regulatory mechanisms.

To further define the role of eIF4G in translation, we determined effects of limiting eIF4G on the translational efficiency of all yeast mRNAs. Translational dependence on eIF4G is not homogeneous across the genome. Limiting eIF4G availability reduces the translational of genes that encode small proteins from transcripts with long poly(A) tails. To test the hypothesis that eIF4G availability influences the poly(A) tail's ability to enhance translation, we determined the translational efficiency of mRNAs with varying poly(A) tail length under wild type and eIF4G-limiting conditions. eIF4G limitation does not alter the relationship between poly(A) tail length and translation, *in vitro*. Therefore, mRNA-specific differences in the requirement for eIF4G are mediated by a poly(A)-independent mechanism.

3.2 Methods

3.2.1 Yeast strains and culture

All yeast cultures are S288C derived. YBC87 is from [166] (*MATa his3 trp1 ura3* L-A-o M-o Gal+). All various eIF4G mutants were created from one of these two parents (see Table 3.1). Yeast cultures were grown in YPAD (1% Yeast extract, 2% Peptone, 0.01% Adenine hemisulfate, 2% Dextrose) at 30°C according to standard protocols [167], unless otherwise noted.

Table 3.1. Yeast Strains used in this study.

Strain	Genotype	Source
YBC87	*MATa his3 tr p1 ura3* L-A-o M-o Gal+	[166]
YBC88	*MATa his3 tr p1 ura3* L-A-o M-o Gal+ *tif4631::HIS5+(Sz. Pombe)*	This study
YBC89	*MATa his3 tr p1 ura3* L-A-o M-o Gal+ *tif4632::HIS5+(Sz. Pombe)*	This study

3.2.2 Microarray analysis.

3.2.2.1 Cell growth, lysis, and RNA isolation

For polysomal RNA samples, yeast cultures were grown to an OD_{600} of 1.0 at 30°C in YPAD and transferred into prechilled centrifuge bottles containing $1/100^{th}$ the culture volume of freshly prepared 10mg/ml cycloheximide in water (final concentration 0.1mg/mL cyloheximide in culture). Cells were harvested (~8Kxg), washed once in 50mLs of cold lysis buffer (20 mM HEPES-KOH, pH 7.4, 2 mM Magnesium Acetate, 100 mM Potassium Acetate, 0.1 mg/ml *fresh* cycloheximide, 3 mM *fresh* dithiothreitol (DTT)), harvested as before, resuspended in 20mLs of cold lysis buffer, transferred to 50ml centrifuge tubes, and harvested at ~5Kxg. Pellets were resuspended in 1.5 mL lysis buffer plus 0.2 U/µl RNasin (Promega) per gram of cell pellet, and then 5 g of cold 0.5mm glass beads per gram of cell pellet was added. Cells were lysed by vortexing for 3 minutes (six 30 second intervals with at least 1 minute on ice between). The crude extracts were spun at 10,000 rpm in a microcentrifuge at 4°C for 20 min, and the resulting supernatant was applied to the gradient immediately or flash-frozen in liquid nitrogen and stored at -80°C for subsequent analysis. 50 OD_{260} units of polysomal extract were applied to an 35-ml 10 to 50% sucrose gradient (prepared in 1X lysis buffer) and spun at 27,000 RPM for 3.5hrs in a Beckmann SW-28 swinging bucket rotor. When measuring the concentration of the lysate, an OD_{260} reading of 100 is equivalent to 100 OD/mL and therefore 0.1mLs would be loaded onto the gradient; lysates were typically 100-200 OD/mL. Polysomal RNA containing portions were isolated (n+2 peak on, see Figure 3.3A) and proteins were removed by adding 2.25 volumes of 8M guanidine hydrochloride and vortexing. To precipitate the RNA, 3.25 volumes of isopropanol was added and the mixture was incubated overnight at -20°C. The precipitated RNA was harvested at 30,000 x *g* for 30min, resuspended in 10mM Tris pH8, 1mM EDTA (TE buffer), and precipitated again by adding 0.1 volumes of 3M Sodium Acetate pH 5.3 and 2.5 volumes of 100% ethanol. RNA was either stored precipitated or harvested as above, resuspended in water, and quantified with a UV spectrophotometer.

For total RNA samples, cells were grown and lysed as described for polysomal RNA isolation. An equivalent volume of phenol pH 6.7 was added to the clarified lysate and the mixture was incubated at 65°C for 10 minutes, vortexing every minute. The samples were cooled on ice for 5 minutes and spun at max speed in a microcentrifuge for 5min. The aqueous layer was transferred to a fresh tube, an equivalent volume of phenol:chloroform:isoamyl alcohol (25:24:1) was added, the mixture vortexed for 4 minutes, and then spun at max speed in a microcentrifuge for 5 minutes. The aqueous layer was transferred to a fresh tube and the RNA precipitated by adding 0.1 volumes of 3M Sodium Acetate pH 5.3 and 2.5 volumes of 100% ethanol. RNA was either stored precipitated or harvested as above, resuspended in water, and quantified with a UV spectrophotometer.

3.2.2.2 cDNA synthesis and labeling, array fabrication, and hybridization

See section 2.2.7.2 for details.

3.2.2.3 Experimental design

To assess ribosome occupancy, cDNA was synthesized from an equivalent amount of polysomal and total RNA isolated from the same strain and each was labeled with a different fluorophore (either Cy5 or Cy3) and hybridized to a single two-color array. To determine relative transcript abundance, cDNA was synthesized from an equivalent amount of total RNA from each strain (wild type, *tif4631*Δ , or *tif4632*Δ). Wild type total RNA from that specific biological replicate was used as a common reference and cDNA from each of the three strains was compared to it directly on a single two-color array.

3.2.2.4 Statistical analysis

Two biological replicates of each strain were analyzed in all experiments. Dye-swap technical replicate pairs were included for each of the biological replicates. Standard pre-processing (manual removal of low quality/physically damaged spots, background correction (normexp method)) and normalization (print-tip loess, intensity based quantile, and ratio based scale) methods were applied [185, 186]. A pooled correlation method was applied to within-array replicate spots, a linear model was fit to the data, and empirical Bayes-based statistical analysis of differential expression was performed (employing Benjamini & Hochberg multiple testing correction) using the limma package from Bioconductor [187-189]. Detailed analysis code is in Appendix I.

3.2.2.5 Raw data

All data is MIAME compliant and the full set of raw and processed data, along with detailed methods regarding data processing have been deposited in the ArrayExpress (http://www.ebi.ac.uk/arrayexpress/) database under accession numbers E-MEXP-2400 and E-MEXP-2402.

3.2.3 Statistical analysis of translationally modified array features

Data for the desired group of genes was extracted from published datasets using the systematic name (not all features (usually non-coding) had data which is why the number of

data points reported is smaller than the number of features in that group). This distribution of numbers was compared to the distribution extracted using the systematic names for all array features above a certain threshold (Figure 3.5A) using the wilcox.test() function in R (http://www.r-project.org/). For associations between poly(A) tail length and Pab1 binding, the number of features in 4 categories (1. Abnormal poly(A) tail/associated with Pab1 and translational efficiency modified, 2. Normal poly(A) tail/ not associated with Pab1 and translational efficiency modified, 3. Abnormal poly(A) tail/associated with Pab1 and no change in translational efficiency, 4. Normal poly(A) tail/ not associated with Pab1 and no change in translational efficiency) were organized into a matrix and assessed with Fisher's Exact Test using the fisher.test() function in R. Code available in Appendix I.

3.2.4 Reverse-transcription quantitative PCR analysis (RT-qPCR)

To prepare total RNA, a 7.5mL yeast culture was grown to mid-log phase (OD_{600} ~= 1.0) in rich media and cells were harvested at 4000g for 5 minutes. Cell pellet was resuspended in 0.5mL of phenol (pH 4.1) by vortexing and 0.5mL of AES buffer (50mM Sodium Acetate pH 5.2, 10mM EDTA, 1% SDS) was added. The samples were heated at 65°C for 10 minutes, vortexing every minute, then cooled on ice for 5 minutes and spun at max speed in a microfuge for 5 minutes. The aqueous layer (~0.5mL) was transferred to a fresh tube and 50µL of 3M sodium acetate pH 5.3 and 550µL of isopropanol were added to precipitate the RNA. The precipitate was harvested at max speed in a microfuge for 30 minutes at 4°C, and the pellet washed with 1mL of 70% ethanol, then dried, and resuspended in water. RNA quality (based on rRNA bands) was inspected on an agarose gel. The sample was treated with 1 unit of RQ1 DNAse (Promega) for every microgram of RNA (as judged by A_{260}) at 37°C for 30 minutes to remove any contaminating DNA. Polysomal RNA was prepared as described above (see Section 3.2.2.1).

5µg of RNA was pre-annealed to a 1:1 mixture of oligo dT and random primers (0.32µg/µL each) by incubation at 70°C for 10min followed by 10min on ice. cDNA was synthesized at 42°C overnight using and Roche Transcriptor (0.75µL/rxn) and deoxynucleoside triphosphates (1mM each A, T, C, and G). Remaining RNA was acid hydrolysed (100mM EDTA, 200mM NaOH, 65°C 10min, neutralize with 0.5M HEPES pH7) and cDNA purified using Qiagen MinElute columns. 1-1.5ng of cDNA was used in qPCR reactions along with 400nM each primer and 1x Fast SYBR Green Master Mix from Applied Biosystems. See Table 3.2 for primer sequences. Following a 1 minute, 95°C enzyme activation step 40 cycles of 95°C, 10 seconds and 60°C, 45 seconds were performed and a melting curve analysis followed. Samples were measured in triplicate and a no reverse transcriptase control was included to detect incomplete genomic degradation. Relative amounts of each transcript were calculated from a standard curve and cDNA samples normalized using *RPL23B* (*SNU13, PST1, MBF1, AGA1, TKL1* and *DOT6)*.

3.2.5 Construction of luciferase reporters with varying poly-adenosine tail length

Various poly(A) tail lengths, generated from synthetic oligonulceotides, were cloned into an existing reporter (from [104]) as SacI and SacII fragments. The plasmid is pUC19-based and has a T7 promoter just upstream of an artificial 18NT 5'-UTR and firefly

luciferase (FF-LUC) gene which is followed by an artificial 46NT 3'-UTR. This vector was digested with SacI and SacII and treated with calf intestinal phosphatase (CIP). Pairs of poly(dT) and poly(dA) oligos, designed with restriction overhangs (see Table 3.2 below), were annealed, phosphorylated with T4 Polynulceotide Kinase (NEB: M0201S) and ligated with T4 DNA ligase (NEB: M0202S) into the digested vector. Positive clones were screened by PCR (with oligos ~50bp up- and down-stream of the insertion site, see Table 3.2) and confirmed by sequencing (see Table 3.3), which gave a close estimate of poly(A) length, which was difficult to determine with precision.

3.2.6. *In vitro* translation

3.2.6.1 *In vitro* transcription of RNA templates

Plasmids were digested with 3 units/µg Ecl136II (Fermentas ER0251) for > 5hrs at 37°C. Digestion was confirmed on a 1% agarose TAE gel, and digested plasmids were phenol extracted and ethanol precipitated, aiming for a concentration of 1-2mg/mL. The transcription reaction contains: 0.1-0.2mg/mL digested plasmid, 30mM Tris-HCL pH8.1, 10mM $MgCl_2$, 2mM spermidine, 0.01% triton-X 100 (these components are preassembled into a 5X mix), 10mM DTT, 1 unit/µL RNAsin (Promega N2611), 2mM ATP, 0.2mM GTP, 2mM CTP, 2mM UTP, 2mM 3'-Ome-m7G(5')ppp(5')GTP (NEB S1411), 0.1 mg/ml T7 polymerase (purified by Kaihong Zhou), 1ng/µL pyprophosphatase (Roche: 10108987001). This reaction was incubated at 37°C for 1.5hrs and 0.1 units/µL of DNAse (RNAse-free; Promega M6101) is added and the mixture incubated at 37°C for another 15min. The RNA is then purified using RNA Clean&Concentrator-25 columns (Zymo Research). RNA was eluted in 35µL HEPES-KOH pH 7.5 (20 mM), distributed into small, single-experiment aliquots and stored at -20°C until use.

3.2.6.2 Translation extract preparation

Cultures derived from a single yeast colony were grown to an OD_{600} between 1 and 1.2 and harvested at 8Kxg for 10 minutes. The cell pellet was resuspended in 15mL of Buffer A (30mM Hepes-KOH pH 7.4, 100mM KOAc, 2mM $Mg(OAc)_2$)+2mM DTT+8.5% mannitol, transferred to a pre-weighed 50mL centrifuge tube and washed 3x10mL with the same buffer. The cell pellet was then weighed and 5 times this mass of 0.5mm glass beads and 1.5 times this mass of the above buffer plus 0.1mM PMSF, 1mg/mL heparin and 1- Roche Protease inhibitor tablet was added. The cells were lysed by vortexing for 5x1minute with 1minute on ice between vortexes. This lysate was spun for 5min at 4Kxg, the supernatant transferred and further clarified by 2x20minute spins at 27Kxg. The clarified supernatant was dialyzed for 2x1.5 hours against 4L (total) of 30mM Hepes-KOH pH 7.4, 100mM KOAc, 2mM $Mg(OAc)_2$, 2mM DTT, 0.1mM PMSF. The dialyzed extract is aliquoted (~200µL), flash frozen in liquid nitrogen and stored at -80°C.

3.2.6.3 Translation reaction and luciferase detection

Initial quantification of reporter RNA concentration was determined by absorbance at 260nm. Concentrations were adjusted based on densitometry measurements of the full-length RNA, run on an ethidium bromide-stained 1% agarose TAE gel. The intensities of

full-length RNA bands were normalized to a lane-by-lane loading control of 40 ng linearized dsDNA plasmid. RNAs for a given translation experiment were run in duplicate on the same 1% agarose gel. In rare cases, RNAs that showed significant smearing or degradation were discarded and transcribed again.

Each 15µL translation reaction consisted of: 2µL of RNA (at the concentration desired for the experiment, based on initial quantification at 260nm), 9.5µL of translation extract, 2.5µL of "6X Mix" (132mM Hepes-KOH, pH7.4, 720mM KOAc, 9mM $Mg(OAc)_2$, 4.5mM ATP, 0.6mM GTP, 150mM Creatine Phosphate, 0.24mM of each amino acid (from 1mM Promega Stock), 10.2mM DTT), 0.5µL Creatine Kinase (10mg/mL in 20mM Hepes-KOH, pH 7.4 50% glycerol), 0.15µL of PMSF (100mM), 0.25µL Roche protease inhibitor (1tab/1mL), 0.1µL RNAsin (40units/µL). 2µL of each RNA stock was added to the appropriate wells of a 96-well RNAse free PCR plate on ice. 13µL of the translation master mix (composed of everything but the RNA) is added to each well, pipetting up and down to mix, keeping the plate on ice. The reactions were spun briefly to collect all material in the bottom of the well and incubated at 26°C for 20min in a 96-well PCR block. After 20 minutes the entire plate is transferred to ice and incubated for 2min to halt reactions simultaneously. 7.5µL of 3X passive lysis buffer is added to each well and the reactions are warmed to room temperature by a 2 minute incubation in the 26°C PCR block. 17.5µL of this reaction is added to 20µL of Steady Glo reagent (Promega E2520) in a round bottom 96 well plate (Corning 3789) and luciferase output read using the Veritas microplate luminometer (Turner BioSystems).

All reactions were run in duplicate, these technical replicates were combined within each experiment and all averaged replicates for each tail length/input amount were normalized to the value for the longest poly(A) tail length/input amount (which consistently gives the highest output). These normalized values are then compared between biological replicates (where reactions are performed with different translation extract and RNA preparations) and this is the error reported. Since the experimental design was similar for many of these reactions an R function was written to facilitate their processing. This code can be found in Appendix I.

3.2.7 RNA hybridization analysis of *in vitro* translated mRNA

in vitro translation reactions (2X volume) were set up as described in Section 3.2.8, however the extract was pre-treated with micrococcal (S7) nuclease to remove endogenous transcripts. Translation extracts were supplemented with 0.5mM $CaCl_2$ and 0.1mg/mL S7 nuclease (Roche 10107921001) and incubated at 26°C for 5min. 2mM EGTA was added to quench the nuclease and the extract was used as described above.

Following incubation and reaction quenching, half of the reaction was assayed for luciferase as described above. To extract RNA from the second half, an equivalent volume of Phenol:Chloroform:IAA was added and the samples were vortexted for 1minute and centrifuged at 13kxg for 5 minutes. 5µL of the aqueous solution was then used in the Nothern Blot below. As a control, 2µL of input RNA was mixed with 13µL of translation extract buffer (30mM HEPES-KOH 7.4, 100mM KOAc, 10mM $Mg(OAc)_2$) and 7.5µL of 3x passive lysis buffer. The RNA was phenol extracted as with the *in vitro* transcribed.

5µL of extracted RNA was added to 14.5µL of RNA loading buffer (66% formamide, 23% formaldehyde(37%), 0.6X MESA (10X = 0.4M MOPS, 10mM NaOAc, 1mM EDTA)). The samples were heated @ 65C for 15min and then 2µL of RNA dye (50% glycerol, 10mM EDTA, pinch of bromophenol blue) was added. The entire sample was loaded on a 1% agarose, 6.5% formaldehyde, 1X MESA gel and run in 1X MESA for ~1.5hrs at 90V. The buffer was continuously mixed with a stir bar to avoid smiling. The gel was soaked 4x5 minutes in 20X SSC and the separated RNA was transferred to a Hybond-N membrane (rinsed in water and then soaked in 20X SSC for 15min) with the capillary method overnight. The membrane was allowed to dry and then cross linked with a Stratagene UV Stratalinker (2X "optimal cross link"). The cross-linked membrane was re-hydrated in 20X SSC and then incubated in 10mL of ExpressHyb (Clontech 636832) rotaing at 65C to block for at least 1hr. During the incubation the a LUC PCR product (generated from OBC356+OBC357) was used to make a body labeled probe using the Rediprime II Labeling Kit (GE RPN1633).The pre-hybridization solution was replaced with 10mLs of fresh ExpressHyb and then 20µL of probe solution was added and incubation continued at 65C for at least 1hr. The hybridization solution was poured down the drain and the membrane was rinsed with 2X SSC, 0.1% SDS (warmed to 65C) and then washed 2x5min with 2xSSC, 0.1%SDS at 65C then 2x5min with 1XSSC, 0.1%SDS at 65C, then 2x5min with 0.1XSSC, 0.1% SDS at 65C. The membrane was then washed with 0.1X SSC (no SDS) at RT, allowed to dry and exposed to a phospho screen.

Table 3.2. Oligonucleotides used in this study.

Ref	Name	Sequence
OBC269	SNU13_1F	AAGAAAGGTGCCAACGAAGCT
OBC270	SNU13_1R	ACAGTCAGCAGCCATAATGATGA
OBC281	PST2_1F	GAGGCAGAAAAGAAGGGAATTG
OBC282	PST2_1R	CAACTTCTGGAGACAACGTTTCCT
OBC273	MBF1_1F	GACAAGAAAATGTCGCAGAAGGA
OBC274	MBF1_1R	GCGGCTTCATAATCGTTTACCA
OBC285	AGA1_1F	ACCAAGACAAACGATGCAAATG
OBC286	AGA1_1R	GCCAGCTTGAACGATAGTGGAT
OBC277	TKL1_1F	CTGCTGGTGCCGTTAGATTGT
OBC278	TKL1_1R	ACACCGATAGAGTCATGTGTAGCAA
OBC289	DOT6_1F	GCACCATAACCTCCGATACCA
OBC290	DOT6_1R	TGCGGGTTTTTGGAATTAGG
OBC301	RPL23B_1F	GTGGATAGCCAAAATGGAAGATG
OBC302	RPL23B_1R	TTTAAGGCCCAGAAGTAGATTTCG
OBC308	SacII_pA100_F1	GGAAAA

OBC365	SacI_pA90_R2	CTTT

3.3 Results

3.3.1 Depletion of eIF4G remodels the yeast transcriptome

Saccharomyces cerevisiae expresses eIF4G isoforms from two genetic loci, *TIF4631* and *TIF4632* [173]. Yeast eIF4G isoforms exhibit extensive functional overlap and, accordingly, only one gene is required for growth (See Chapter 2 and [173, 207]). Tif4631 is three times more abundant than Tif4632 and *tif4631*Δ strains contain significantly reduced levels of eIF4G, the extent of which correlates with growth and translation initiation defects (See Chapter 2 and [191, 207]). On the other hand, *TIF4632*-deletion modestly reduces eIF4G levels and has no detectable phenotypes [207]. Therefore, *tif4631*Δ and *tif4632*Δ strains can be used to examine the effects of significant and mild eIF4G-depletion, respectively.

To determine the message specific effects of eIF4G gene depletion we compared genome-wide transcript abundance between *tif4631*Δ (YBC88) or *tif4632*Δ (YBC89) and wild type strains (YBC87) using two color microarrays (Figure 3.1A; Section 3.2.2; raw data is available at ArrayExpress accession number: E-MEXP-2400). Strong reproducibility between strain replicates illustrates the high quality of this data (Figure 3.2).

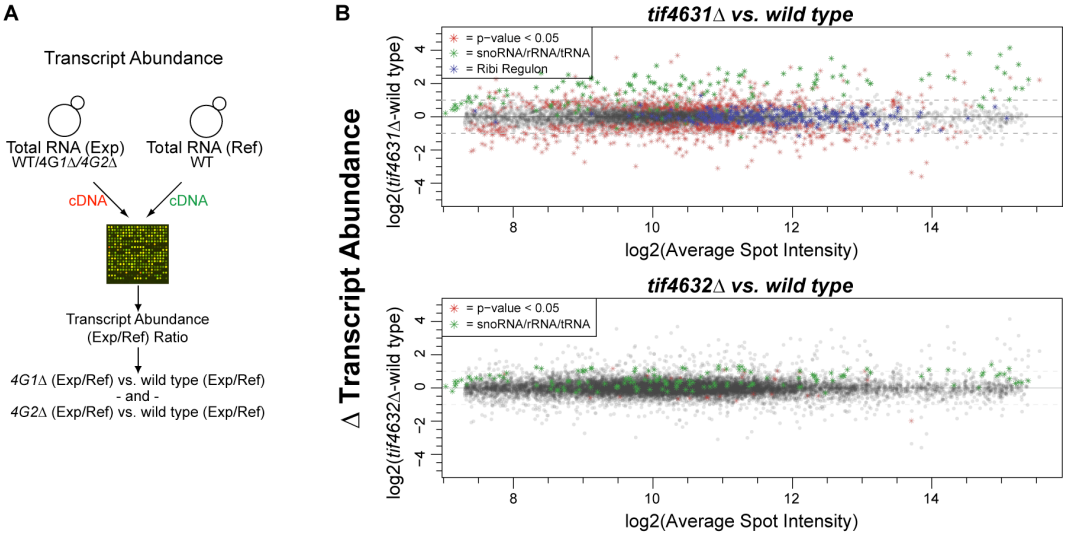


Figure 3.1. Effect of eIF4G gene deletions on genome-wide transcript abundance. (**A**) Schematic outlining how changes in relative transcript abundance were determined. WT = wild type, 4G1Δ = *tif4631*Δ, 4G2Δ = *tif4632*Δ, Exp = experimental sample, Ref = common reference used in all hybridizations. (**B**) Plots displaying differences in transcript abundance ($\log_2$(4GΔ(red/green)) - $\log_2$(wild type(red/green)); vertical axis) versus average spot intensity (0.5($\log_2$(4GΔ(red+green)) + $\log_2$(wild type(red+green))); horizontal axis) between *tif4631*Δ (top) or *tif4632*Δ (bottom) and wild type strains. Statistically significant changes in transcript abundance are indicated by red "*"s. All expressed snoRNA/tRNA/rRNA and Ribi regulon [208] features are indicated by green and blue "*"s, respectively. Dashed horizontal lines indicate a 2-fold ($\log_2$(1)) change.

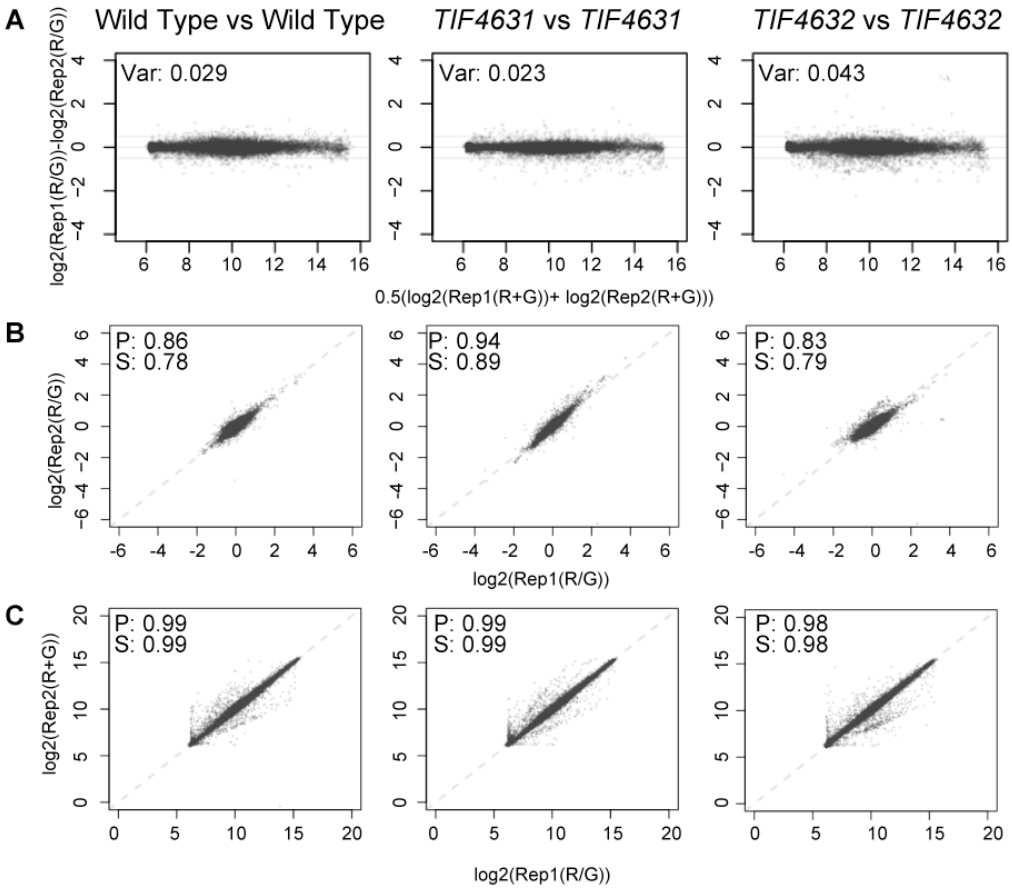


Figure 3.2. Reproducibility of transcript abundance microarray datasets. (**A**) M values (log2(R/G); R = red intensity, G = green intensity) of dye swap technical replicates were sign corrected and averaged for each biological replicate (Rep1 and Rep2) and differences in these values (log2((Rep1a(R/G)-Rep1b(G/R))/2)-log2((Rep2a(R/G)-Rep2b(G/R))/2)) are plotted (vertical axis) against average spot intensity (0.5(log2((Rep1(R+G)+Rep2(R+G)))); horizontal axis). Variances (Var) of these comparisons are indicated on the plot. A tight vertical distribution around zero is indicative of high similarity between biological replicates. M values (**B**) and average spot intensities (**C**) of biological replicates are plotted against each other and the Pearson (P) and Spearman rank (S) correlation of these comparisons is indicated.

Deletion of *TIF4631* significantly modified the transcript abundance of ~12% of the genome (869 genes (435 up, 434 down); Figure 3.1B, red "*"s). Genes involved in ribosome biogenesis (GO ID: 42254) and translation elongation (GO ID: 6414) were enriched amongst the induced group (p-value = 1.34e-05 and 2.8e-04, respectively) [121, 209]. Inspection of the genes assigned to these categories revealed that snoRNAs and tRNAs constitute greater than 88% (47/53) and 97% (43/44) of the two lists, respectively. In fact, more than 70% (47/67) of expressed snoRNA array features, 90% (48/52) of expressed tRNA features, and 50% (5/10) of expressed rRNA features have significantly (p-value < 0.05) increased transcript abundance in *tif4631*Δ cells (Figure 3.1B, green "*"s). Consistent with these findings, *TOR1*, which is essential for transcription of ribosome components [210], displays increased relative transcript abundance (1.6-fold; p-value < 1.1223e-04) in *tif4631*Δ cells. On the other hand, transcripts from only 18 of the 207 expressed ribosome biogenesis (Ribi) Regulon genes [208] are strongly increased, which is

a significant depletion among *tif4631*Δ-affected mRNAs (Fisher's exact test p-value < 0.04)(Figure 3.1B, blue "*"s). Therefore, the deletion of *TIF4631* induces a very specific component of the ribosome biogenesis response. This is consistent with a recent study implicating *TIF4631* in ribosome biogenesis and its genetic and physical interactions with other ribosome biogenesis factors [211-215].

In addition to upregulation of certain ribosome biogenesis factors, depletion of eIF4G reduces transcripts from genes related to amino acid biosynthesis (GO ID 8652; p-value 7.28e-10) and protein folding (GO ID 6457; p-value 3.87e-3). This could reflect a decreased demand for these proteins as a result of reduced growth and translation initiation rates in the *tif4631*Δ strain.

Modestly decreasing eIF4G levels, via deletion of *TIF4632*, significantly modified the transcript levels of less than 1% of the genome (32 genes (19 up, 13 down). The list of genes whose transcript level decreased in *tif4632*Δ cells was not enriched in any particular biological process. On the other hand, genes related to translation elongation (7 tRNA features) and ribosome biogenesis (3 snoRNA and 1 rDNA feature) were induced, similar to *tif4631*Δ cells (Fig. 3.1B). In fact, the vast majority (17 of 19) of genes that are upregulated in *tif4632*Δ cells also display significantly increased abundance in *tif4631*Δ cells. This suggests the existence of an isoform-independent response to reduction in overall eIF4G levels and is consistent with the extensive functional overlap of yeast isoforms observed previously (See Chapter 2).

3.3.2 *TIF4631*-deletion reduces translation of short mRNAs with long poly(A) tails

To further characterize the cellular response to eIF4G depletion, we examined the fraction of transcripts engaged in translation (ribosome occupancy [26]) for all genes expressed during log phase growth in wild type (YBC87), *tif4631*Δ (YBC88) and *tif4632*Δ (YBC89) strains. Ribosome occupancy was determined by comparing the relative mRNA abundance in equivalent amounts of polysomal and total RNA using two-color microarrays (Figure 3.3A left; raw data is available at ArrayExpress accession number: E-MEXP-2399). The polysome/total (P/T) ratio of each gene was then compared between strains and assessed for statistically significant differences (See Methods Section 3.2.2.3 for details). Comparison of replicate (within strain) P/T ratios and spot intensities illustrates the high reproducibility of the microarray data, which is similar to previous reports (Figure 3.4A, [216]).

Deletion of *TIF4632* did not significantly modify the ribosome occupancy of any mRNAs from their wild type state (Figure 3.3B bottom). In contrast, the *tif4631*Δ strain contained a number of genes whose P/T ratio was significantly (p-value < 0.05) modified from its wild type level (Figure 3.3B top). To further characterize these genes, they were first split into two groups: (1) those whose $tif4631\Delta_{P/T}$ > wild $type_{P/T}$ (enhanced ribosome occupancy; 217 array features) and (2) those whose $tif4631\Delta_{P/T}$ < wild $type_{P/T}$ (repressed ribosome occupancy; 199 array features). Data from a recently curated set of gene, mRNA

and protein characteristics [217] were compiled for both groups and the distributions were compared to that of all genes that were expressed under the conditions examined using a Mann-Whitney U test (7384 distinct probes; Figure 3.5). Overall, the distributions of mRNA abundance, protein abundance and protein decay rates for genes with enhanced ribosome occupancy are similar to those for genes expressed during log phase growth (p-value > 0.1; Table 3.4). Genes whose translational efficiency was reduced as a consequence *of TIF4631* deletion are highly expressed at the message and protein level, and encode more stable proteins (p-value < 0.04; Table 3.4).

Interestingly, statistical analysis revealed a positive correlation between ribosome occupancy and open reading frame length. ORFs whose ribosome occupancy was reduced in *tif4631*Δ cells were significantly shorter than ORFs whose ribosome occupancy was either increased or unaffected by *tif4631*Δ (p-value = 1.9e-6; Table 3.4). Recent genome-wide investigations found that ORF and poly(A) tail lengths are inversely correlated in yeast [218]. Given that eIF4G contributes to translational stimulation by poly(A) tails via interaction with the poly(A) binding protein [156], we examined the relationship between ribosome occupancy in our data and poly(A) tail length (using poly(A) tail length data from [218]). The group of transcripts that were preferentially lost from polysomes in *tif4631*Δ cells were enriched for long, and devoid of short poly(A) tail containing messages (p < 2.1e-3, Fisher's exact test, Table 3.4). Genes whose ribosome occupancy was enhanced in the *tif4631*Δ strain were not any more or less likely to have a short poly(A) tail; however, having a long poly(A) tail decreased the chances that a message would shift into polysomes in the absence of *TIF4631* (p-value = 1.2e-4; Table 3.4).

Since the functional interaction of eIF4G with poly(A) tails is mediated by the poly(A) binding protein Pab1 [156], and poly(A) tail length is positively correlated with Pab1 binding [218], we examined correlations between ribosome occupancy in *tif4631*Δ and Pab1-association, as determined by co-immunoprecipitation (IP) microarray analysis [219]. In accordance with the negative association between ribosome occupancy and poly(A) tail length, transcripts that strongly associate with Pab1 show reduced polysome occupancy in the absence of *TIF4631* (p-value = 1.19e-8; Table 3.4). Transcripts whose polysome association increased in the *tif4631*Δ strain were not more or less likely to be found associated with Pab1 (p-value > 0.05; Table 3.4). Therefore, deletion of *TIF4631* specifically compromises the ribosome association of small transcripts with long poly(A) tails that associate with Pab1.

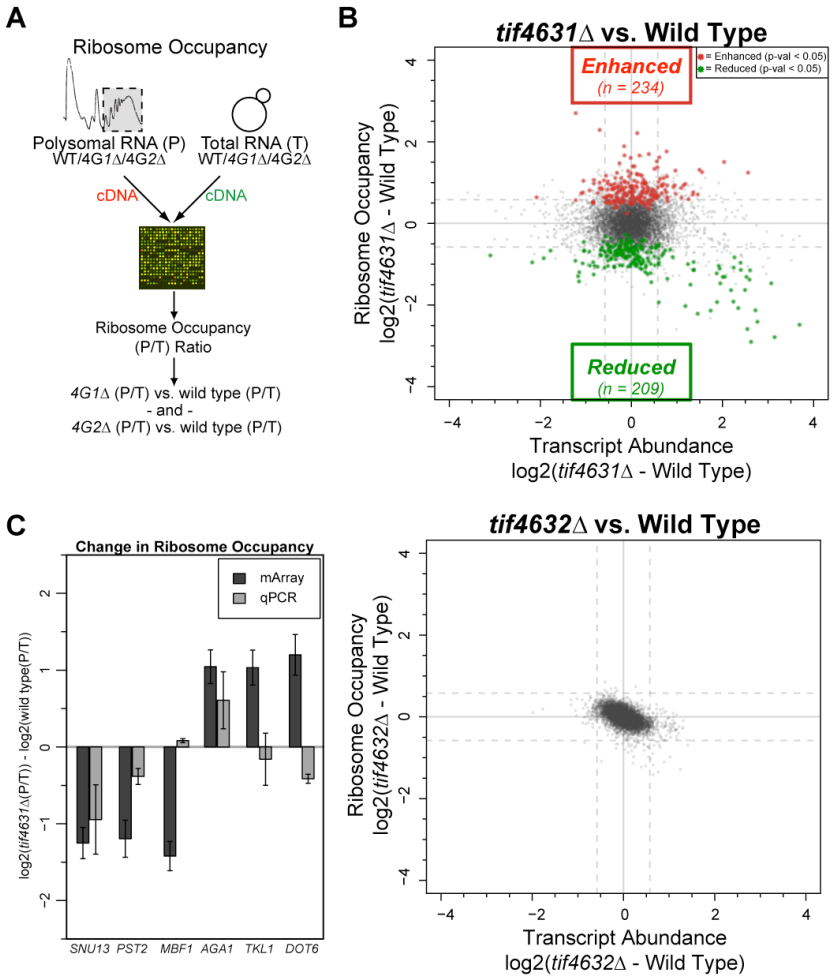


Figure 3.3. Genome-wide analysis of ribosome occupancy in wild type and eIF4G deletion strains. (**A**). Schematic of how differences in ribosome occupancy (the fraction of transcripts engaged in translation (P/T)) were determined (for details see Section 3.2.2). WT = wild type, *4G1*Δ = *tif4631*Δ, *4G2*Δ = *tif4632*Δ, (**B**) Plots displaying differences in ribosome occupancy ($\log_2$(*mutant*(P/T)) - $\log_2$(wild type(P/T)); vertical axis) and transcript abundance ($\log_2$(*mutant*(Exp/Ref)) - $\log_2$(wild type(Exp/Ref))horizontal axis) between *tif4631*Δ (top) or *tif4632*Δ (bottom) and wild type cells. Array probes with significant (p value < 0.05) enhanced and reduced ribosome occupancy are indicated by red and green "*"s, respectively. Dashed lines indicate 1.5-fold ($\log_2$(1)) changes in the respective direction. Axis lines are solid.

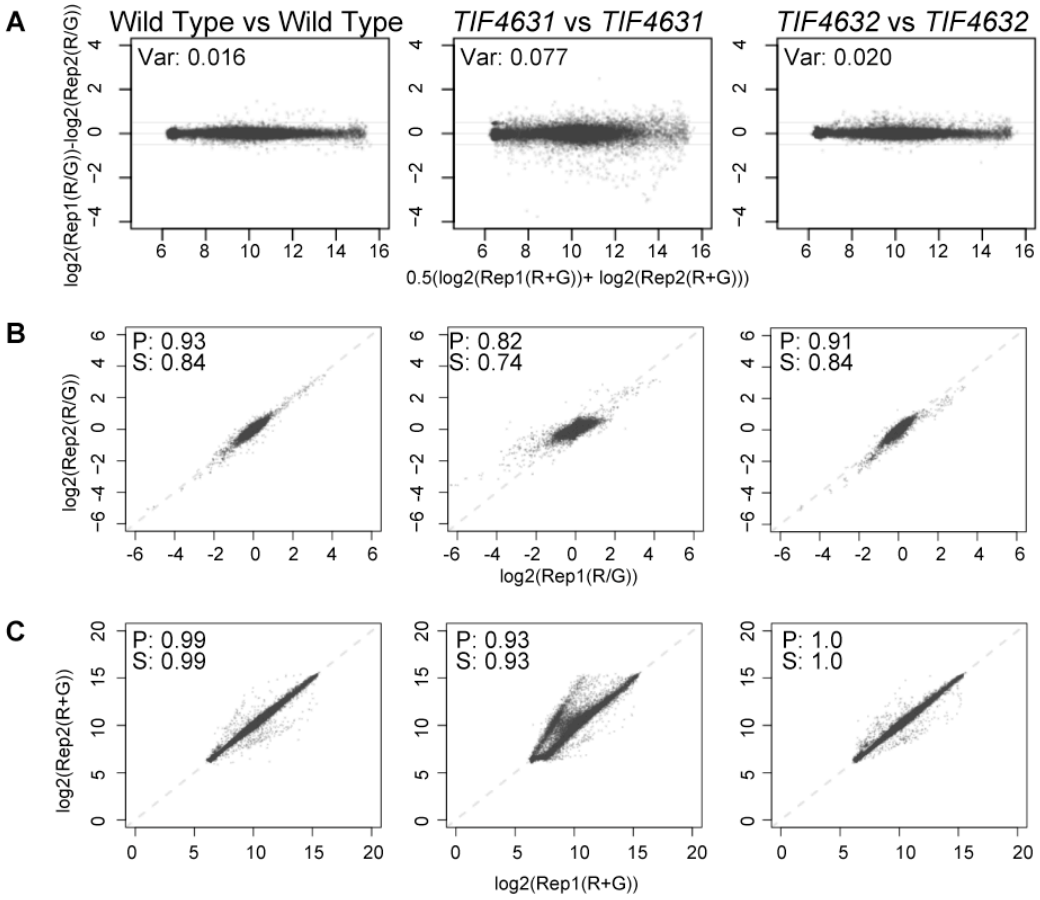


Figure 3.4. Reproducibility of ribosome occupancy microarray datasets. (**A**) M values (log2(R/G); R = red intensity, G = green intensity) of dye swap technical replicates were sign corrected and averaged for each biological replicate (Rep1 and Rep2) and differences in these values (log2((Rep1a(R/G)-Rep1b(G/R))/2)-log2((Rep2a(R/G)-Rep2b(G/R))/2)) are plotted (vertical axis) against average spot intensity (0.5(log2((Rep1(R+G)+Rep2(R+G))))); horizontal axis). Variances (Var) of these comparisons are indicated on the plot. A tight vertical distribution around zero is indicative of high similarity between biological replicates. M values (**B**) and average spot intensities (**C**) of biological replicates are plotted against each other and the Pearson (P) and Spearman rank (S) correlation of these comparisons is indicated.

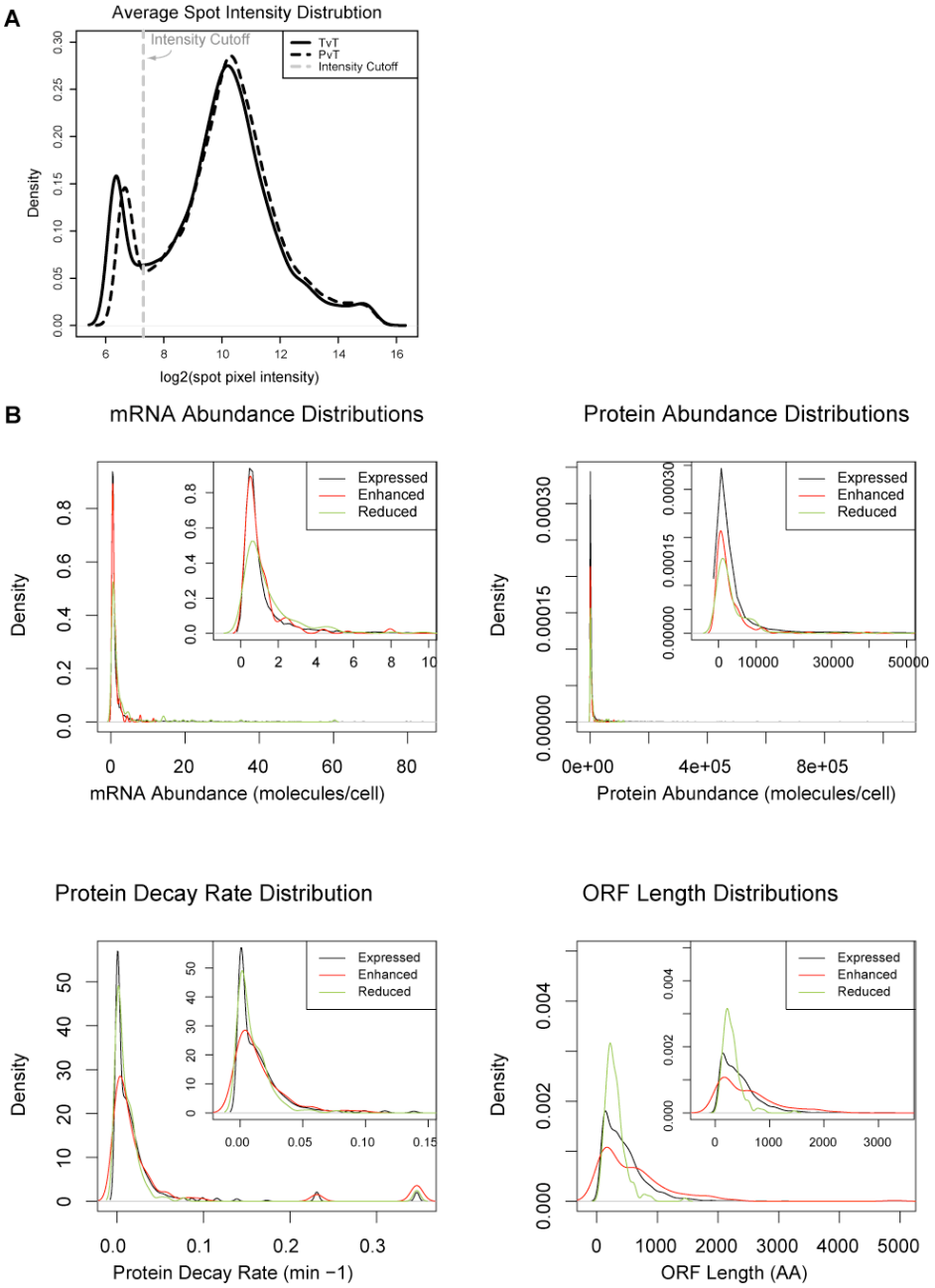


Figure 3.5. Microarray spot intensity and mRNA characteristic distributions of translationally modified genes (**A**) Kernel density plot displaying the distribution of average spot intensities for all array probes in mRNA abundance (TvT) or ribosome occupancy (PvT) data sets. This plot was used to determine an appropriate intensity cutoff (indicated as vertical dashed line) that distinguishes signal from background. (**B**) Kernel density plots displaying the distributions of the indicated characteristic for the two groups of genes whose ribosome occupancy was significantly different in *tif4631*Δ cells (enhanced in red, reduced in green) as well as the background distribution for all probes above the intensity cutoff (black). Inset plot is zoom to better show peaks containing the majority of the data.

Table 3.4. Characteristics of translationally modified features

	Expressed		Ribosome Occupancy Enhanced			Ribosome Occupancy Repressed		
mRNA characteristics*	Median	(n)	Median	(n)	p-value	Median	(n)	p-value
mRNA abundance (copies/cell)	0.7	(5419)	0.7	(141)	0.9	0.9	(135)	1.35e-4
Protein abundance (copies/cell)	1456	(5448)	1539	(143)	0.8	1887	(137)	0.029
Protein decay rate (-min)	0.01	(4163)	0.01	(107)	0.16	0.006	(111)	0.031
ORF length (amino acids)	376	(5639)	492	(150)	0.063	274	(142)	1.85e-6
poly(A) tail association**	Percent Associated	(Assoc/ Total)	Percent Associated	(Assoc/ Total)	p-value	Percent Associated	(Assoc/ Total)	p-value
PASTA-Short	9.6	(480/4980)	11.2	(15/133)	0.55	2.4	(3/127)	2.06e-3
PASTA-Long	14.0	(701/4980)	3.7	(5/133)	1.22e-4	25.2	(32/127)	6.80e-4
Pab1 associated	48.2	(1948/4040)	46.6	(55/118)	0.068	64.4	(73/113)	1.19e-8

*Data from a curated list of message characteristics [217] was compiled for genes that displayed significantly (p-value < 0.05) enhanced or reduced ribosome occupancy in *tif4631*Δ versus wild type cells. The distributions were compared to that of all expressed features using the Mann-Whitney U-test (p-values are indicated). The number of features in each group with data is indicated (n).

**Lists of genes that displayed significantly (p-value < 0.05) enhanced or reduced ribosome occupancy in *tif4631*Δ cells as compared to wild type were examined for enrichment of short and long poly(A) tail containing [218] and Pab1 associated [219] messages. Significance of associations between ribosome occupancy and poly(A) tail length and Pab1 binding were determined using Fisher's exact test (p-values indicated).

Of the six open reading frames that encode messages with abnormally long poly(A) tails and whose P/T ratio was at least 2-fold different between *tif4631*Δ and wild type cells, we were able to confirm changes in ribosome occupancy for two (SNU13 and PST2) of three (MBF1 showed no differential expression) using reverse-transcription quantitative PCR (RT-qPCR) (Figure 3.3C). One of three ORFs with strongly (> 2-fold) enhanced ribosome occupancy following *tif4631*Δ was confirmed by RT-qPCR (Figure 2.9C). Given the subtle effects (most changes were < 2-fold) of *TIF4631*-deletion on global translation, the biological significance of these effects will need to be confirmed in downstream experiments (see below).

3.3.3 Translational efficiency correlates positively with poly(A) tail length *in vitro*

The inverse relationship between poly(A) tail length and changes in translational efficiency under limiting eIF4G conditions suggests that eIF4G availability might influence the ability of the poly(A) tail to act as a translational enhancer. To test this directly, we determined the translational efficiency of reporter mRNAs with varying poly(A) tail lengths in translation extracts made from wild type and eIF4G-limited strains (i.e. *tif4631*Δ) (Figure 3.6A). We constructed a panel of twelve luciferase (LUC) reporters, with poly(A) tails ranging from 0 to 100 nucleotides, which represents the range found *in vivo* (see Table 3.3 and section 3.2.5)[218, 220]. Utilizing the longest tail message, we identified a concentration

that minimizes saturation of the extract by longer tail messages and allows detection of the less potent, shorter tail reporters (100ng/reaction; Figure 3.6B).

Multiple (n=4) wild type extract preparations exhibit a strong positive relationship between poly(A) tail length and translational efficiency, consistent with previous reports (the average is shown in Figure 3.6C) [220]. All reporters exhibit a similar half-life over the timeframe of the assay, ruling out differences in mRNA stability as the cause of differences in translational efficiency observed. Alteration of poly(A) tail length is also unlikely based on a previous study that found poly(A) tail length to be maintained in similar experiments [220]. As shown in Figure 3.6E, the relationship between poly(A) tail length and translation was similar in extracts from wild type and *tif4631*Δ strains, suggesting that eIF4G does not significantly influence poly(A) tail-mediated translational enhancement. Therefore, limiting eIF4G levels modulates the translational efficiency of mRNAs by a mechanism independent of poly(A) tail length.

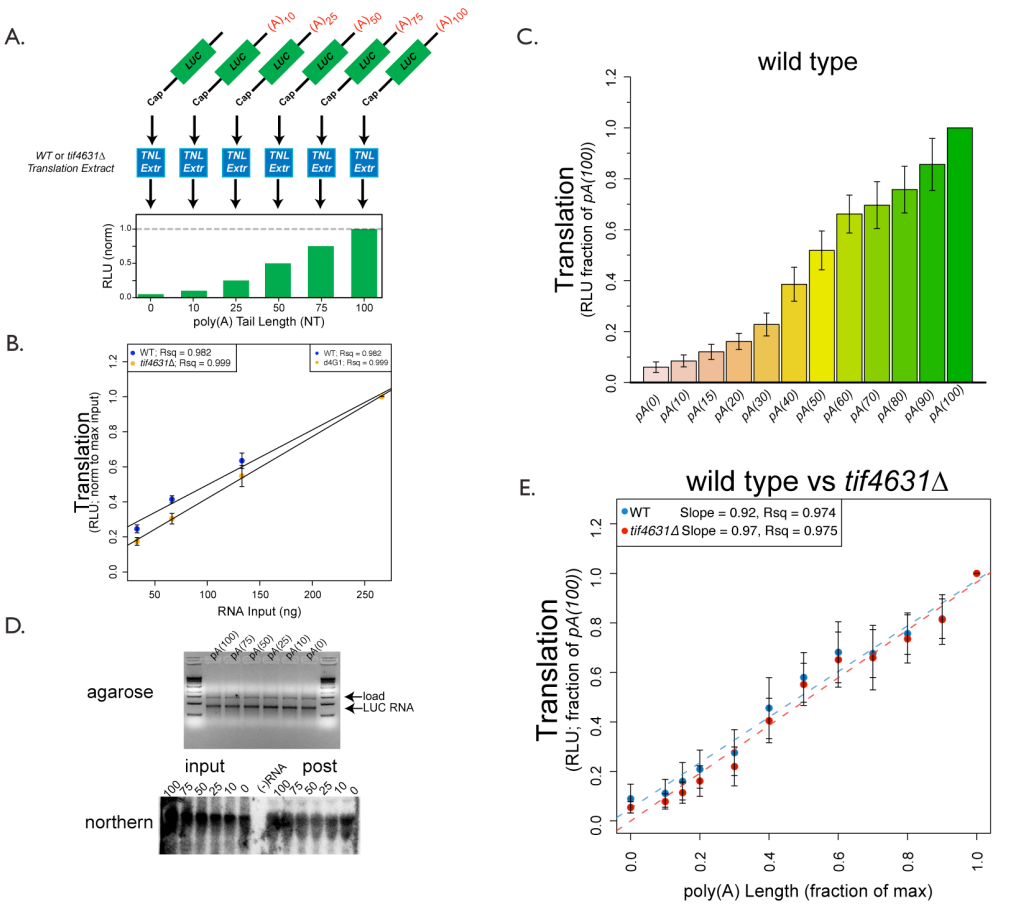


Figure 3.6. Translation as a function of poly(A) tail length in wild type and eIF4G-limited extracts. (**A**) Schematic of experimental design. A series of luciferase (LUC)-based reporters was constructed (only a fraction is depicted, see Table 3.3 and horizontal axis of (B) for complete list), and used to produce 7-methyl guanosine capped RNA with varying poly(A) tail lengths *in vitro*. Each of these RNAs was used to program translation extract from wild type or *tif4631*Δ strains and luciferase production was assayed. For each reporter, translational output (in relative light units (RLUs)) was measured in duplicate, averaged and normalized to that with the longest (100NT) poly(A) tail. Normalized values from multiple RNA (n=2) and extract (n = 4) preparations were averaged with the error indicated. (**B**) RNA titration. Translational output (in RLU) from varying amounts of pA(100) reporter RNA (indicated on horizontal), was normalized to the highest input, averaged between extract preparations (n=4; error indicated), and fit to a linear regression (R-squared (Rsq) shown). (**C**) Translation of 100ng of the indicated RNA was assayed in wild type extract as described in (A). (**D**) Stability of LUC reporters in translation extracts. Input RNA is shown on an agarose gel (above (LUC RNA)), along with a DNA loading control (load) used for input normalization. An equivalent amount of this (input) was phenol extracted along side RNA used in an *in vitro* translation extract (post) and quality assessed by Northern Blot (bottom) with a luciferase-specific probe. (**E**) Translation as a function of poly(A) tail length was determined in wild type (blue) and *tif4631*Δ (orange) extracts as described in (A). 100ng of each RNA was input and the poly(A) tail length values normalized to the longest value facilitate interpretation of linear fit.

3.4 Discussion

Limiting eIF4G availability is a conserved translational regulatory mechanism. For example, nutrient deprivation triggers eIF4G degradation in yeast and both apoptosis and viral infection result in targeted cleavage of mammalian eIF4G, all of which coincide with a decrease in global translation rates [9, 14, 206]. To further elucidate the consequences of eIF4G limitation, we determined the message-specific effects of limiting eIF4G availability.

Taking advantage of the large functional overlap and differential expression that exists between the two yeast eIF4G isoforms, we utilized *tif4631*Δ and *tif4632*Δ strains to examine the effects of significant and modest eIF4G-depeletion, respectively. Severe eIF4G limitation induces a very specific component of ribosome biogenesis machinery, increasing the message abundance of 50% of ribosomal RNAs, greater than 70% of snoRNAs and more than 90% of tRNAs expressed during wild type growth (Figure 3.1). Simultaneously, amino acid biosynthetic transcript abundance is reduced. These effects mimic the bacterial response to decreased translation initiation rates [87, 221-224], and suggest an evolutionarily conserved mechanism of coping with translation-inhibiting perturbations amongst these microbes.

Perhaps the most striking finding from our study is that the translation of a narrow range (~5%) of yeast mRNAs is sensitive to reduced overall eIF4G levels (Figure 3.3). This suggests that dependence on eIF4G is not homogenous across the genome. Recent studies have redefined the role of eIF4G in translation initiation, bringing the necessity of closed loop formation into question and identifying cellular messages that can recruit ribosomes in a eIF4G-independent manner [85, 90]. Global statistical evaluation illustrates that *TIF4631* deletion does not affect translational efficiency for the vast majority of transcripts (compare Figures 3.7 and 3.4). This is consistent with the small range of effects on ribosome occupancy observed here, the overall modest effects of eIF4G depletion on genome wide translational efficiency observed recently and reports that mammalian translation rates are only mildly sensitive to eIF4G reduction [163, 204, 225].

Nonetheless, depletion of eIF4G significantly modifies the ribosome occupancy of particular messages (Figure 3.3). Among the mRNAs whose translational efficiency is reduced, a significant fraction contain unusually long poly(A) tails and hence are likely to bind to more copies of Pab1 (Table 3.4, [218])). A positive correlation between eIF4G level and translational efficiency of genes whose translation is expected to be more Pab1-dependent supports previous studies showing that Pab1-mediated, poly(A) tail driven stimulation of translation *in vitro* requires eIF4G [156]. Previous work has highlighted the translational regulatory potential of gene-specific differences in poly(A) tail length in yeast [218]. Our results suggest that regulation of eIF4G levels and activity could influence the effects of the poly(A) tail on mRNA translation. To examine this further we determined the relationship between poly(A) tail length and translational efficiency under wild type and eIF4G limiting conditions. We observed a strong positive correlation between translation

and poly(A) tail length and found that limiting eIF4G availability had no influence on this relationship *in vitro* (Figure 3.6). Despite the statistically significant relationship between poly(A) tail length and sensitivity to eIF4G reduction, only ~25% of the transcripts exhibiting reduced ribosome occupancy had long poly(A) tails. Moreover, the translation of some mRNAs with short or average length poly(A) tails was also affected by decreased eIF4G levels, suggesting other mechanisms of regulation are influenced by eIF4G availability.

Given eIF4G's influence on the activity of other initiation factors [65, 66, 76, 79, 86, 87] and its inherent RNA binding activity [226], other properties such as longer 5'-untranslated regions (UTRs) [163], affinities for the cap-binding protein eIF4E, or particular sequence elements could also contribute to eIF4G sensitivity. No significant relationship between UTR length, as determined in three recent studies [227-229], and eIF4G level sensitivity could be detected (Figure 3.8). This is inconsistent with a recent study [85] that observed an increased average 5'-UTR length for genes whose translation was enhanced by reduction in eIF4G levels. This discrepancy could stem from utilization of different 5'-UTR length datasets. However, a single gene encodes a diverse population of yeast messages with varying 5'-UTRs and this could explain the discrepancies between 5'-UTR lengths from several recent reports (Figure 3.8) [227]. Examining effects of eIF4G depletion on individual mRNA species within these populations would be a better way to examine the relationships between 5'-UTR properties and eIF4G sensitivity, but is beyond the scope of this study. Interestingly, both this and another recent report [85] observed a strong positive correlation between open reading frame length and changes in translational efficiency under eIF4G-limiting conditions. Given the recent discovery of a relationship between ORF length and the stability of the cap-eIF4GF interaction this will be an interesting avenue to explore [230].

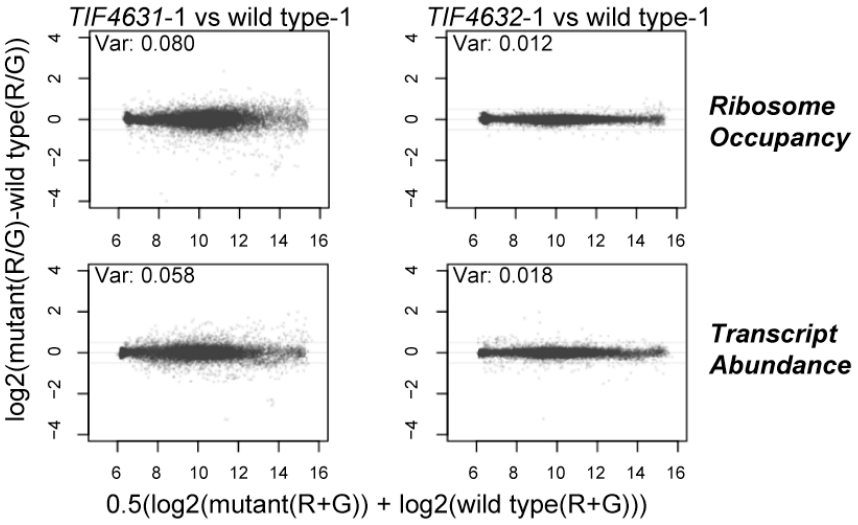


Figure 3.7. Global variation of message abundance and ribosome occupancy. Differences in M values (log2(R/G); R = red intensity, G = green intensity) between strains are plotted (vertical axis) against average spot intensity (log2((Rep1(R+G)+Rep2(R+G))/2); horizontal axis) as in Figures 3.2A and 3.4A. Variances (Var) of these comparisons are indicated on the plot.

Genetic depletion of eIF4G was recently found to narrow the range of translational efficiencies in yeast [225]. The distribution of translational efficiencies for the transcripts whose ribosome occupancy is enhanced by deletion of *TIF4631* is statistically indistinguishable from the distribution of all expressed genes (p-value > 0.98). However, transcripts whose translational efficiency was reduced in the *tif4631*Δ strain exhibited significantly increased translational efficiency in the wild type strain (p-value 2.7e-6), suggesting that deletion of *TIF4631* partially reduces the range of genome wide translational efficiency.

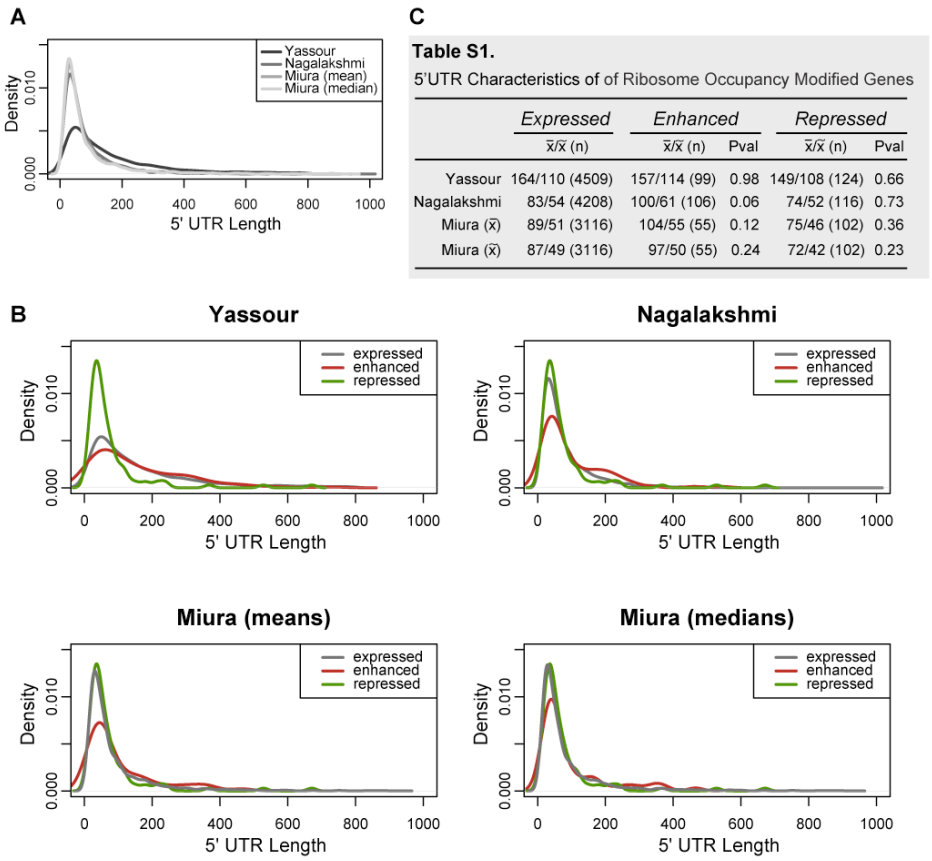


Figure 3.8. 5'-untranslated region analysis of translationally modified genes. (**A**) Kernel density plot displaying the distribution of 5'-untranslated region (UTR) lengths for all expressed genes (average spot intensity above the background cutoff; see Figure 3.5A) from 3 published data sets [227-229]. One study [227] provides several UTR lengths for each gene and either the mean (Miura (mean (x bar))) or median (Miura (median(x tilde))) of all values was utilized. (**B**) Kernel density plots displaying the distributions of UTR lengths from the indicated study for the two groups of genes whose ribosome occupancy was significantly different in *tif4631*Δ cells (enhanced in red, reduced in green) as well as the background distribution for all probes above the intensity cutoff (dark gray). (**C**) Statistical assessment (Wilcoxon rank sum/Mann-Whitney U test) of differences between the enhanced or repressed and expressed distributions plotted in (B).

Exquisite dependence on eIF4G concentration for translation initiation may result from reduced eIF4G binding affinity by an mRNA, or from more complex indirect effects on translation factors and ribosome affinity. In either case, changes in eIF4G expression are sufficient to trigger changes in the production of specific proteins, providing potentially important regulation of cellular activity. These findings reveal the existence of a class of eIF4G-dependent mRNAs that are uniquely sensitive to overall eIF4G levels.

Chapter 4

Function of the hepatitis C virus internal ribosome entry site in *Schizosaccharomyces pombe*

4. Function of hepatitis C virus internal ribosome entry site function in *Schizosaccharomyces pombe*.

4.1 Introduction

The hepatitis C virus (HCV) is a major public health problem. Over 180 million people are infected worldwide, making HCV a leading cause of chronic liver disease [112]. HCV infections are extremely persistent and current treatments are associated with undesirable side effects and exhibit limited efficacy [113, 114]. The development of new therapies relies on a mechanistic understanding of essential components of the HCV life cycle. Translation of the HCV genome produces proteins required for viral replication and therefore represents a good treatment target. Non-specific inhibition of translation would be toxic but, fortuitously, HCV RNA initiates translation in a non-canonical manner that can be preferentially targeted. Unlike most eukaryotic mRNAs, a sequence in the 5'-untranslated region of the HCV genome directly recruits and positions the small ribosomal subunit internally, and therefore is referred to as an internal ribosome entry site (IRES) [121, 122]. Accordingly, translation of the HCV RNA requires only a small number of host factors (eIF2, eIF3 and eIF5/5B) for efficient initiation [44].

The 5'-UTR of HCV RNA contains five domains of considerable secondary structure, four of which assemble into a tertiary fold to form the IRES [134]. A number of biochemical and assembly experiments have mapped interaction surfaces and assigned translational functions to each of the domains (see Chapter 1 and [231]). The influence of host factors on IRES function is still being determined. For example, a recent study identified a eukaryotic-specific ribosomal protein (Rps25) that is required for efficient initiation by both the cricket paralysis (CrPV) and hepatitis C viral IRESs but, interestingly, has no detectable function in canonical initiation. The authors took advantage of yeast genetics to conclusively demonstrate the requirement of Rps25 in cricket paralysis virus (CrPV) translation, but could not do so for the HCV IRES, whose function in *Saccharomyces cerevisiae* is unclear. Attempts have been made to characterize IRES function in budding yeast but have been hindered by a lack of detectible activity [232, 233]. Other studies have reported HCV function in *S. cerevisiae*, however the low activities detected cannot be unambiguously assigned to bona fide IRES-mediated translation [234-236].

The lack of a genetically tractable system has hindered identification and analysis of host factors influencing HCV IRES function. To this end, we determined the ability of the HCV IRES to mediate translation in *S. pombe* both *in vivo* and *in vitro*. Preliminary findings suggest additional components are required for efficient HCV IRES-mediated translation in fission yeast and current work is focused on identifying these factors. Establishing a genetic system to understand mechanisms of HCV translation will guide therapeutic design and illuminate mechanisms of translational control.

4.2 Methods

4.2.1 Yeast strains and culture

Haploid S. pombe cells (*h- ura4-D18 leu1-32 ade2-216 his3-D*; a generous gift from Fei Li in the Cande Lab) were either grown in rich (YES: 0.5% yeast extract, 3% glucose and 225mg/L adenine, histidine, leucine, uracil and lysine hydrochloride) or selective (PMG (-Ura): see http://www-bcf.usc.edu/~forsburg/media.html for recipe) media according to standard protocols.

4.2.2 Plasmid construction

An optimized GFP (S65T) was PCR amplified (with OBC241+225) from the pFA6a-GFP(S65T)-kanMX6 vector and inserted into pREP2 (pBC33) as a SalI-XmaI fragment to generate pBC34 (REF: Heim Nature 1995; see http://www-bcf.usc.edu/~forsburg/vectors.html for details on the backbone vectors). pKB84 (a.k.a pBC35; see Berry 2010 for details) contains a T7 promoter for the production of HCV IRES (NT 40-372)-luciferase (LUC) RNA *in vitro* (see below). The G267C IRES mutant was created from this construct using QuickChange mutagenesis (with OBC235+236) to give pBC36. The wild type and mutant IRES were PCR amplified (with OBC247+248) from these two vectors and inserted into pBC33 as a SalI fragment to give pBC42 and pBC43.

4.2.3 GFP visualization

Various GFP expressing plasmids were transformed by electroporation according to standard methods (see http://www-bcf.usc.edu/~forsburg/tfmn.html for details) and positive transformants isolated on selective media (PMG(-Ura)). Colonies were picked and grown to mid-log phase in selective media, a small volume of which was placed on a coverslip and examined qualitatively by fluorescence microscopy.

4.2.4 Immunoblot analysis

Strains were grown to ~0.25 OD600, a T = 0 aliquot was removed and then shifted to 48°C to induce heat shock. At indicated timepoints, 10mL of the culture was harvested by centrifugation at 4Kxg for 10 minutes. The pellet was washed with 5mL of STOP buffer (150mM NaCl, 10mM EDTA, 50mM NaF, 1mM NaAzide) and harvested again as before. The cells were resuspended in 1mL of STOP buffer, transferred to an eppendorf, briefly pelleted, and the buffer removed by aspiration. 100µL of RIPA buffer (25mM Tris 8, 150mM NaCl, 0.1% NP40, 0.1% SDS, 12mM NaDOC, 1mM PMSF, 1/10th a Roche Complete protease inhibitor tablet) and 200µL of glass beads (0.5µM) were added to the pellet and the cells were lysed by bead beating (6x 30sec with 1min on ice between). The lysate was clarified by a 20min spin at 13Kxg and the supernatant transferred to a fresh eppendorf. Absorbance at 280nm was measured and different lysates adjusted to the same concentration. SDS sample buffer was added to 1X, the samples boiled for 5min and then loaded onto a 10% SDS-PAGE gel, which was run at 225V until the dye reached the bottom. The proteins were transferred to a polyvinylidene difluoride membrane (Immobilon-P (Millipore IPVH07850)) at 110V for 1 hour and 25 minutes. The membrane was blocked in 4% milk, 1X-TBS-T (20 mM Tris-HCl, pH 7.4, 500 mM NaCl,

0.05% Tween 20) for 30min at room temperature and then hybridized with the primary antibody (Santa Cruz mouse monoclonal anti-GFP (sc-9996)) at 1:500 in 4% milk, 1X-TBS-T. The antibody solution was removed with two rinses followed by 4 washes (5min each) with 10mLs of 1X TBS-T. The secondary antibody (Amersham anti-mouse-HRP NA9310-1ML) was incubated at 1:2000 for 1hr at room temperature in 1X TBS-T and removed as with the primary. The blot was developed using Amersham's ECL Advance Western Blotting Detection Kit.

4.2.5 *In vitro* translation

4.2.5.1 *In vitro* transcription of RNA reporters

Templates were digested with 3 units/µg EcI136II (Fermentas ER0251) for > 5hrs at 37°C. Digestion was confirmed on a 1% agarose TAE gel, and digested plasmids were phenol extracted and ethanol precipitated, aiming for a concentration of 1-2mg/mL. The transcription reaction contains: 0.1-0.2mg/mL digested plasmid, 30mM Tris-HCL pH8.1, 25mM MgCl2, 2mM spermidine, 0.01% triton-X 100 (these components are preassembled into a 5X mix), 10mM DTT, 1 unit/µL RNAsin (Promega N2611), 5mM ATP/GTP/CTP/UTP, 0.1 mg/ml T7 polymerase (purified by Kaihong Zhou), 1ng/µL pyprophosphatase (Roche: 10108987001). This reaction was incubated at 37oC for 1.5hrs and 0.1 units/µL of DNAse (RNAse-free; Promega M6101) was added and the mixture incubated at 37°C for another 15min. The RNA is then purified using RNA Clean&Concentrator-25 columns (Zymo Research). RNA was eluted in 35µL HEPES-KOH pH 7.5 (20 mM), distributed into small, single-experiment aliquots and stored at -20°C until use.

4.2.5.2 Extract preparation

Cultures derived from a single yeast colony were grown to 1-1.2 OD600 and harvested at 8Kxg for 10 minutes. The cell pellet was resuspended in 15mL of Buffer A (30mM Hepes-KOH pH 7.4, 100mM KOAc, 2mM Mg(OAc)2)+2mM DTT+8.5% mannitol, transferred to a pre-weighed 50mL centrifuge tube and washed 3x10mL with the same buffer. The cell pellet was then weighed and 5 times this mass of 0.5mm glass beads and 1.5 times this mass of the above buffer plus 0.1mM PMSF, 1mg/mL heparin and 1-Roche Protease inhibitor tablet was added. The cells were lysed by vortexing for 5x1minute with 1minute on ice between vortexes. This lysate was spun for 5min at 4Kxg, the supernatant transferred and further clarified by 2x20minute spins at 27Kxg. The clarified supernatant was dialyzed for 2x1.5 hours against 4L (total) of 30mM Hepes-KOH pH 7.4, 100mM KOAc, 2mM Mg(OAc)2, 2mM DTT, 0.1mM PMSF. The dialyzed extract is aliquoted (~200µL), flash frozen in liquid nitrogen and stored at -80°C.

4.2.5.3 *In vitro* translation and measurement of luciferase activity

Initial quantification of reporter RNA concentration was determined by absorbance at 260nm. Concentrations were adjusted based on densitometry measurements of the full-length RNA, run on an ethidium bromide-stained 1% agarose TAE gel. Sodium dodecyl sulfate (SDS,0.025%) was present in the 2x formamide loading dye and was essential for

RNAs to run as clean bands in non-denaturing agarose gels. The intensities of full-length RNA bands were normalized to a lane-by-lane loading control of 40 ng linearized dsDNA plasmid. RNAs for a given translation experiment were run in duplicate on the same 1% agarose gel. In rare cases, RNAs that showed significant smearing or degradation were discarded and transcribed again.

Each 15µL yeast translation reaction consisted of: 2µL of RNA (at the concentration desired for the experiment, based on initial quantification at 260nm), 9.5µL of translation extract, 2.5µL of "6X Mix" (132mM Hepes-KOH, pH7.4, 720mM KOAc, 9mM Mg(OAc)2, 4.5mM ATP, 0.6mM GTP, 150mM Creatine Phosphate, 0.24mM of each amino acid (from 1mM Promega Stock), 10.2mM DTT), 0.5µL Creatine Kinase (10mg/mL in 20mM Hepes-KOH, pH 7.4 50% glycerol), 0.15µL of PMSF (100mM), 0.25µL Roche protease inhibitor (1tab/1mL), 0.1µL RNAsin (40units/µL).

Each 15µL RRL translation reaction consisted of: 2µL of RNA (at the concentration desired for the experiment, based on initial quantification at 260nm), nuclease treated RRL (56% (v/v)), amino acids (20 mM), RNasin Plus RNase inhibitor (1.3U/ml; Promega), Complete Protease Inhibitor Cocktail (Roche: 1 tablet in 1 ml used as 50x),DTT (2 mM), and 1.8 mM MgCl2, 45 mM KCl, and 26 mM KOAc, to obtain final salt concentrations of 2.2 mM Mg2+, 45 mM KCl, and 90 mM KOAc (Kozak 1990; Berry et al.2010).

2µL of each RNA stock was added to the appropriate wells of a 96-well RNAse free PCR plate (ABgene Cat #AB-2800) on ice. 13µL of the translation master mix (composed of everything but the RNA) is added to each well, pipetting up and down to mix, keeping the plate on ice. The reactions were spun briefly to collect all material in the bottom of the well and incubated at 26°C for 20min in a 96-well PCR block. After 20 minutes the entire plate is transferred to ice and incubated for 2min to halt reactions simultaneously. 7.5µL of 3X passive lysis buffer is added to each well and the reactions are warmed to room temperature by a 2 minute incubation in the 26oC PCR block. 17.5µL of this reaction is added to 20µL of Steady Glo reagent (Promega E2520) in a round bottom 96 well plate (Corning 3789) and luciferase output read using the Veritas microplate luminometer (Turner BioSystems).

All reactions were run in duplicate, these technical replicates were combined within each experiment and all averaged replicates for each input amount were normalized to the value for the highest input (which consistently gives the highest output). These normalized values are then compared between biological replicates (where reactions are performed with different translation extract and RNA preparations) and this is the error reported.

Table 4.1. Oligonucleotides used in this study.

Ref	Name	Sequence
OBC241	SalI_GFP-f	ACGCGTCGACGAGTAAAGGAGAAGAACTTTTCACTGG
OBC225	XmaI_GFP-r	TCCCCCCGGGTTATTTGTATAGTTCATCCATGCC
OBC247	SalI_HCVIRES-f	ACGCGTCGACCTCCCCTGTGAGGAAC
OBC248	SalI_HCVIRES-r	ACGCGTCGACTTTTTTCTTTGAGGTTCAGGATTTGTGC
OBC235	HCV_G267C_qc1f	CTGCTAGCCGAGTAGTGTTGCGTCGCGAAAGGCCTTGTGG
OBC236	HCV_G267C_qc1r	CCACAAGGCCTTTCGCGACGCAACACTACTCGGCTAGCAG

Table 4.2. Plasmids used in this study.

Ref	Description
pBC33	pREP2
pBC34	Pnmt1-GFP
pBC35	T7-HCV IRES-LUC
pBC36	T7-HCV IRES(G267C)-LUC
pBC42	Pnmt1-HCV IRES-GFP
pBC43	Pnmt1-HCV IRES(G267C)-GFP

4.3. Results and Discussion

4.3.1 *Schizosaccharomyces pombe* encodes a more mammalian-like eIF3 than *Saccharomyces cerevisiae*

As mentioned above, the HCV IRES requires eIF3 to assemble translation competent ribosomal complexes [44]. eIF3 is a large protein complex consisting of 13 subunits in mammals, 6 of which are conserved in *Saccharomyces cerevisiae* and 10 in *Schizosaccharomyces pombe* (Table 4.3, [38]). The fission yeast and human proteins exhibit a range of homologies that are broadly distributed across each subunit (Figure 4.1). Null mutations in three fission yeast subunits (eIF3d, e and h) are functionally complemented by their human homologs [237-240]. Therefore, expression of a mammalian-like eIF3 complex makes *S. pombe* a good candidate to support HCV IRES-mediated translation.

4.3.2 The Hepatitis C virus IRES cannot efficiently initiate translation in *Schizosaccharomyces pombe*, *in vivo*

To determine the ability of the HCV IRES to mediate translation in *S. pombe*, we introduced reporters where GFP expression relies on either cap- or IRES-dependent translation and monitored expression by fluorescence microscopy (Figure 4.2A). For the cap-dependent reporter, GFP production is under control of a strong endogenous promoter (from the NMT1 gene)[241, 242]. To report on IRES-mediated translation, we inserted the HCV IRES (NT 40-372) just upstream of GFP in the cap-dependent reporter, creating an N-terminal fusion with the first 13 amino acids of the HCV polyprotein (MSTNPEPQRKKST). To monitor non-IRES-mediated translation derived from the IRES-containing construct, we generated a third reporter with a single point mutation (G267C) in the IRES sequence that reduces activity to 2% of wild type [134]. Using the same promoter for all three constructs ensures production of consistent amounts of RNA.

Transformation with the cap-dependent reporter produces robust GFP expression, with greater than 95% of cells exhibiting strong GFP fluorescence. Conversely, only a small fraction (less than 5%) of cells containing the wild type and mutant IRES plasmids fluoresce weakly (Figure 4.2B; data not shown). To more quantitatively assess population wide expression we determined GFP production in mid-log phase cultures by western blot. Cultures containing the cap-dependent reporter express GFP robustly, while both wild type and mutant IRES reporters result in similarly low levels of GFP expression, suggesting it is derived from non-IRES mediated translation such as ribosomal read-through or the production of truncated RNAs by cryptic promoting or splicing events (Figure 4.2B; T = 0). We cannot rule out inactivation of GFP by the addition of a small number of HCV encoded amino acids to the N-terminus however, given its extensive use as a C-terminal fusion, this is unlikely [243].

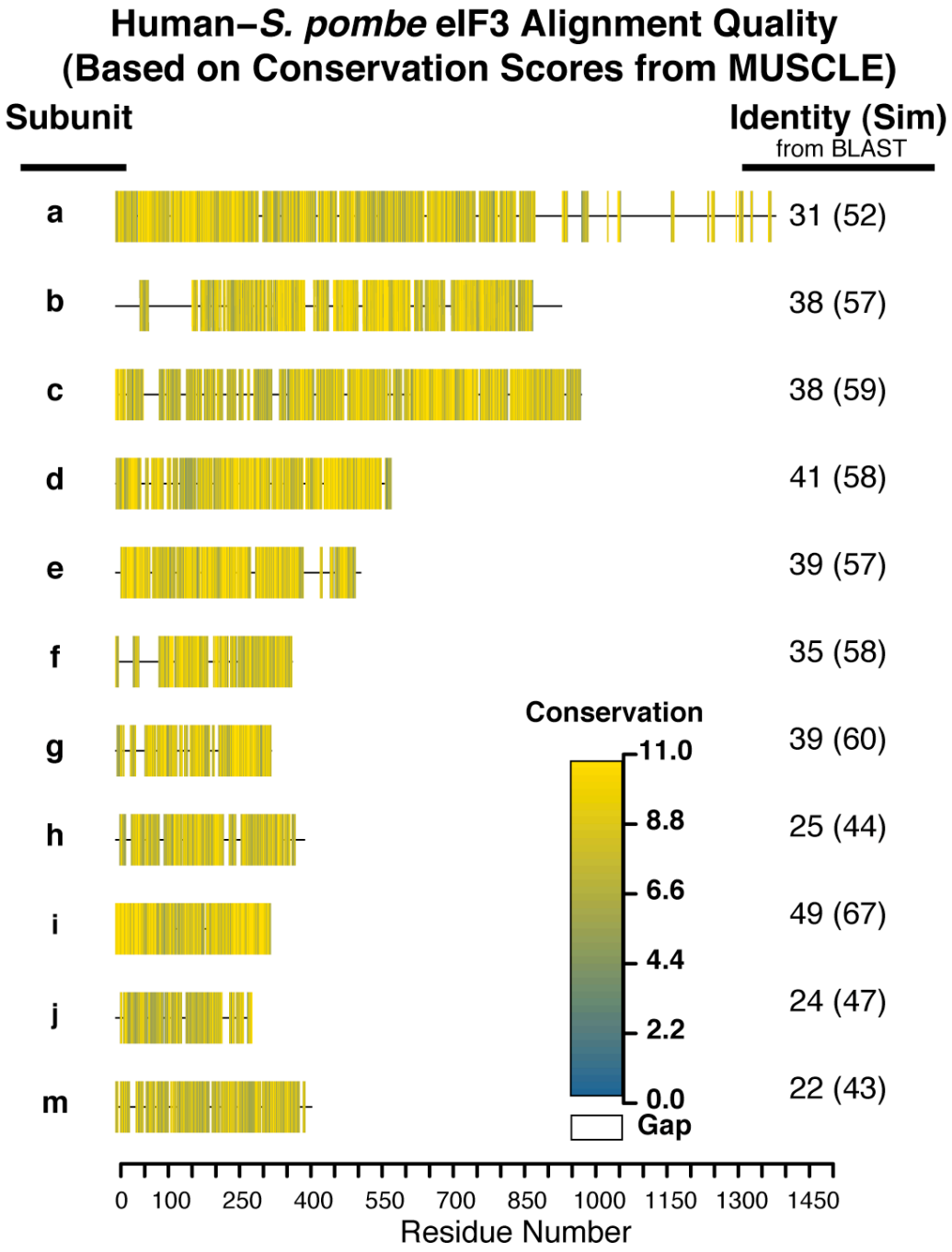


Figure 4.1. Comparison of conserved human and fission yeast eIF3 subunits. A pairwise alignment of human and fission yeast eIF3 subunit protein sequences was generated using MUSCLE [182]. Alignment quality (based on conservation of physico-chemical properties [244]) is indicated by a heat map, where yellow is high quality, blue is low quality, and white is a gap (where sequence in the consensus originates from a single isoform; key on lower right). Black lines represent the length of the alignment (see scale in amino acids on bottom). Global domain homologies are indicated as a percentage of amino acid identity (similarity) to the right of each heat map.

Table 4.3. Summary of eIF3 subunits from selected eukaryotes.

Unified nomenclature	*H. sapiens* Name	*S. cerevisiae* Name	Iden(Sim)	*Sz. pombe* Name	Iden(Sim)	Notes
eIF3a	p170	TIF32	25(46)	p107	31(52)	*S. cer* not complemented by *Sz. pom* or *H. sa*
eIF3b	p116	PRT1	31(53)	p84	38(57)	
eIF3c	p110	NIP1	29(50)	p104	38(59)	
eIF3d	p66	-		Moe1	41(58)	*Sz. pom* null functionally complemented by *H. sa*
eIF3e	p48	-		Int6	39(57)	*Sz. pom* null functionally complemented by *H. sa*
eIF3f	p47	-		Csn6	35(58)	
eIF3g	p44	TIF35	33(53)	TIF35	39(60)	*S. cer* not complemented by *H. sa*
eIF3h	p40	-		p40	25(44)	*Sz. pom* null functionally complemented by *H. sa*
eIF3i	p36	TIF34	44(65)	Sum1	49(67)	*Sz. pom* not functionally complemented by *H. sa*
eIF3j	p35	HCR1		-		
eIF3k	p28	-		-		
eIF3l	p67	-		-		
eIF3m	GA17	-		Csn7B		

Adapted from (REF: Hinnebusch Trends 2006). *S. cer* = *Saccharomyces cerevisiae*, *Sz. pom* = *Schizosaccharomyces pombe*, *H. sa* = *Homo sapiens.* [237-240, 245]

Competition for translation components by endogenous messages could be inhibiting IRES-mediated translation in these experiments. Given that HCV infection induces eIF2α phosphorylation, which coincides with an inhibition of host, but not viral protein synthesis, this is a strong possibility [148]. To mimic the conditions that typify an HCV infection we sought to induce eIF2α phosphorylation in *S. pombe*. Fission yeast encodes three eIF2α kinases, GCN2, which is induced by amino acid starvation, and HRI1/HRI2, which are homologs of the heme-regulated inhibitor kinase and have are induced by arsenic compounds and heat shock [246]. To induce eIF2α phosphorylation, we heat shocked *S. pombe* cultures and then monitored GFP expression from cap and IRES-dependent reporters over time by western blot. Exposure to heat shock for times that far exceed that required for HRI-mediated eIF2α phosphorylation (15min) does not affect IRES-mediated translation (Figure 4.2C [246]). Both the wild type and mutant IRES constructs produce similarly low levels of GFP at levels observed in the absence of heat shock. Minor differences in relative GFP expression between wild type and mutant IRESs varied slightly from experiment to experiment and are difficult to assess due to the lack of a loading control. The consistent level of GFP expression from the cap-dependent reporter throughout heat shock exposure can most likely be attributed to the high stability of this protein, whose half-life far exceeds the timeframe of our experiment [247]. Based on previous results we expect high levels of eIF2α phosphorylation and coincident inhibition of protein synthesis after 15 minutes of heat shock, however we did not assay either of these things directly and it is possible that host translation was maintained at levels sufficient to outcompete IRES-mediated translation.

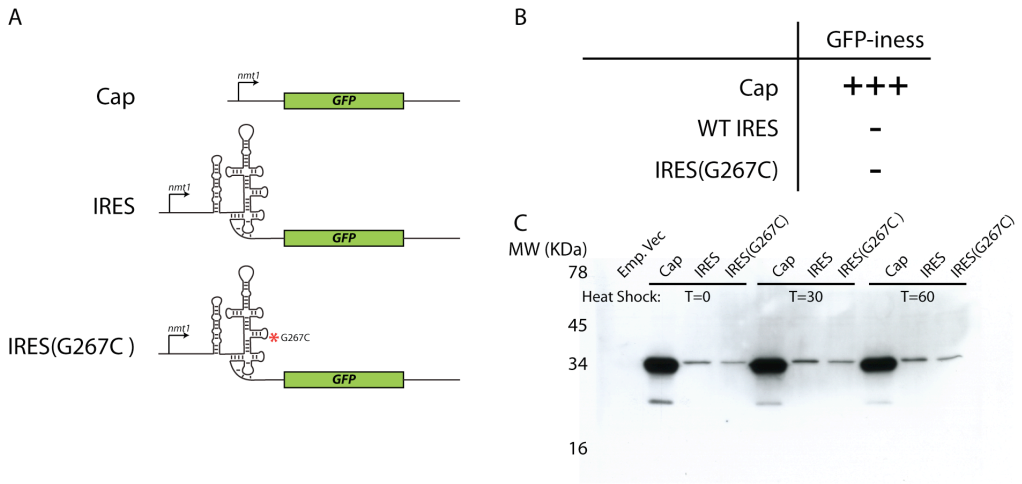
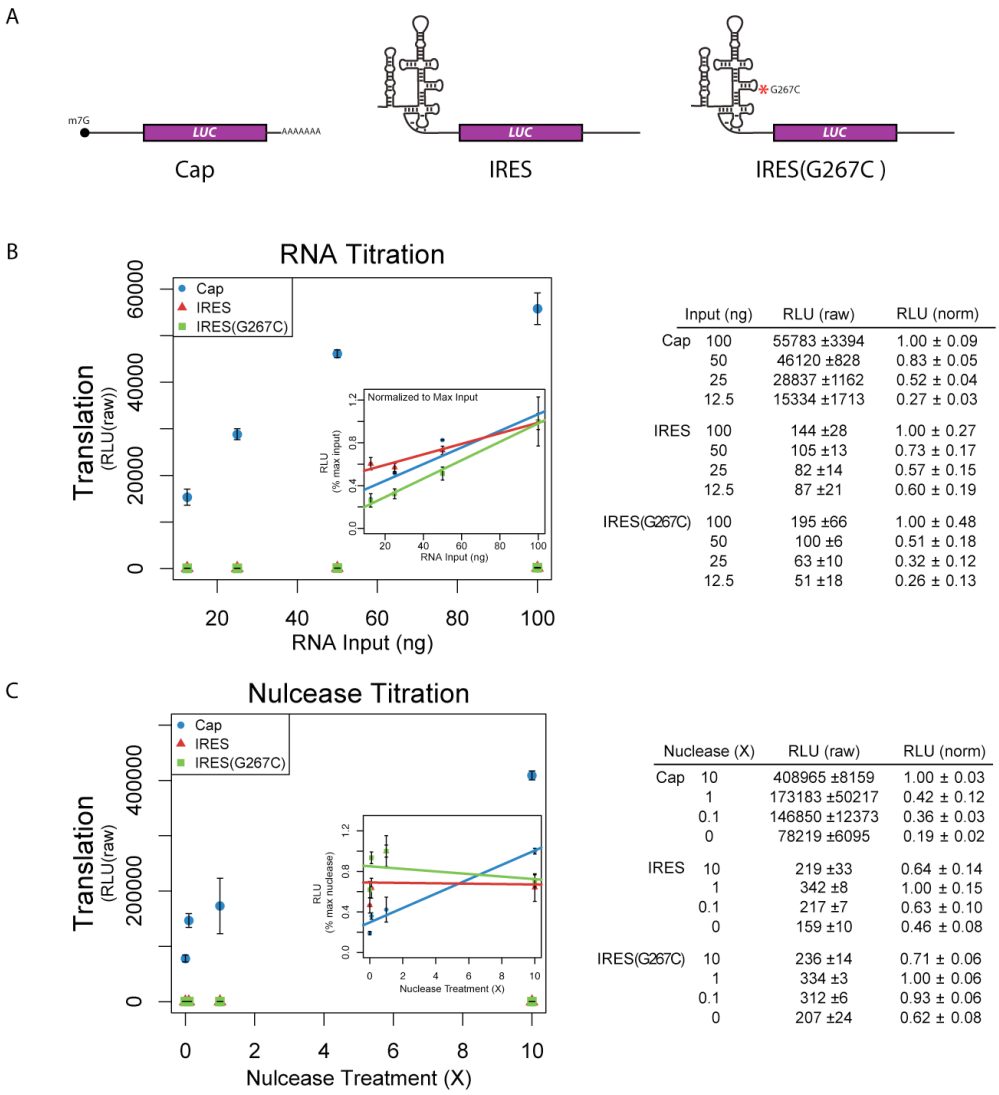


Figure 4.2. Hepatitis C virus internal ribosome entry site function in *S. pombe in vivo*. (**A**) Schematic of the GFP plasmid reporters. (**B**) Qualitative assessment of GFP expression in cells containing each of the reporters in (A) as determined by fluorescence microscopy. (**C**) Western blot determining GFP expression from each reporter throughout heat shock (48°C) treatment.

4.3.3 The Hepatitis C Virus IRES cannot efficiently initiate translation in *Schizosaccharomyces pombe, in vitro*

To rigorously remove competition for translation components, we utilized an *in vitro* translation system where host messages can be eliminated by micrococcal nuclease treatment. First we monitored the effect of increasing RNA concentration on translation following a nuclease treatment regiment utilized in similar extract based systems (See Chapter 3, [248]). Translation of a capped luciferase RNA exhibits a strong positive response to increased input (Figure 4.3A). On the other hand, even at concentrations where the extract becomes limiting for cap dependent translation, uncapped HCV IRES RNA is poorly translated (Figure 4.3A). Moreover, similar levels of translation are produced from an RNA containing the non-functional G267C mutant, suggesting the modest translation observed is not IRES-mediated. To ensure that we were sufficiently removing competition from host messages, we titrated nuclease treatment (arbitrarily setting the treatment used in previous experiments as 1X). Increased nuclease treatment (10-fold) strongly induces cap-dependent translation, but has little effect on HCV IRES-mediated translation (Figure 4.3B). Based on previous studies in HCV IRES-competent translation extracts (e.g. rabbit reticulocyte lysate (RRL)), we expect translation from cap and IRES-dependent RNAs to be on the same order of magnitude [249]. Taken together, these results suggest that *S. pombe* requires additional factors to efficiently promote HCV IRES translation.



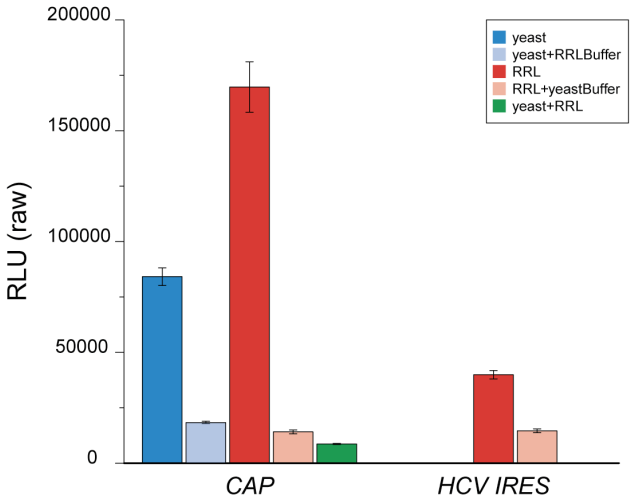
	Input (ng)	RLU (raw)	RLU (norm)
Cap	100	55783 ±3394	1.00 ± 0.09
	50	46120 ±828	0.83 ± 0.05
	25	28837 ±1162	0.52 ± 0.04
	12.5	15334 ±1713	0.27 ± 0.03
IRES	100	144 ±28	1.00 ± 0.27
	50	105 ±13	0.73 ± 0.17
	25	82 ±14	0.57 ± 0.15
	12.5	87 ±21	0.60 ± 0.19
IRES(G267C)	100	195 ±66	1.00 ± 0.48
	50	100 ±6	0.51 ± 0.18
	25	63 ±10	0.32 ± 0.12
	12.5	51 ±18	0.26 ± 0.13

	Nuclease (X)	RLU (raw)	RLU (norm)
Cap	10	408965 ±8159	1.00 ± 0.03
	1	173183 ±50217	0.42 ± 0.12
	0.1	146850 ±12373	0.36 ± 0.03
	0	78219 ±6095	0.19 ± 0.02
IRES	10	219 ±33	0.64 ± 0.14
	1	342 ±8	1.00 ± 0.15
	0.1	217 ±7	0.63 ± 0.10
	0	159 ±10	0.46 ± 0.08
IRES(G267C)	10	236 ±14	0.71 ± 0.06
	1	334 ±3	1.00 ± 0.06
	0.1	312 ±6	0.93 ± 0.06
	0	207 ±24	0.62 ± 0.08

Figure 4.3. Hepatitis C virus internal ribosome entry site function in *S. pombe in vitro*. (**A**) Schematic of the *in vitro* transcribed RNA reporters. (**B**) RNA titration in *S. pombe* translation extracts. Raw translational output (vertical) from varying amounts of luciferase reporter RNAs (see (A); horizontal). Due to the broad dynamic range of experimental values, the data was normalized to the max input within each reporter and plotted (see inset; lines generated from linear regression analysis). Raw and normalized values are indicated on right. (**C**) Nuclease treatment titration. Translation (vertical) from a single input amount of each reporter (see (A)) was assayed in extracts that were treated with micrococcal nuclease to varying degrees (horizontal). To aid in comparison, the data was normalized within each reporter to the value with max nuclease treatment (see inset; lines generated from linear regression analysis). Raw and normalized values are indicated on the right.

Addition of factors to stimulate non-canonical translation in extracts is not a novel concept. Early studies of poliovirus IRES-mediated translation in rabbit reticulocyte lysate reported initiation at numerous nonauthentic internal start codons and found that utilization of the authentic start codon could be maximized by addition of extract from poliovirus-infected HeLa cells (Dorner et al J Vir 1984). To investigate the possibility that mammalian factors could stimulate HCV IRES activity in yeast cells, translation extracts from yeast and rabit reticulocytes were mixed. Both yeast and rabbit reticulocyte lysate translate capped RNA efficiently (Figure 4.4). Mixing equivalent volumes of the two extracts significantly reduced output from the capped RNA (10 and 20-fold decreases from yeast and RRL extracts alone, respectively). Diluting either extract with an equivalent volume of the buffer constituents of the other reduces translation to a similar level, suggesting that either the buffer conditions are not compatible or the relationship between translation and extract concentration is non-linear. Translation mediated by the HCV IRES could only be detected in RRL and was roughly 25% of that produced by an equivalent amount of capped RNA. Two-fold dilution of the RRL with yeast buffer reduced translation nearly linearly (37%), however addition of an equivalent volume of yeast extract completely abolishes and IRES-mediated translation. This could result from the presence of an inhibitor factor or the formation of non-functional heterogenous complexes [233]. Other extract ratios were tried (9:1 and 4:1 yeast:RRL), however there was no detectible signal from the HCV IRES at these higher yeast:RRL mixtures. Fractionation of mammalian extracts might isolate inhibitory from stimulatory factors, and are the focus of ongoing research.

In our hands, the HCV IRES cannot efficiently promote translation in *S. pombe in vivo* or *in vitro*, consistent with reports that failed to detect activity in S. cerevisiae [232, 233]. The extract system established here is a powerful tool to explore the ability of mammalian factors to stimulate HCV IRES-mediated translation in *S. pombe*. Once identified, heterologous expression of these stimulatory factors will create a genetically tractable system to identify and characterize viral and host elements influencing HCV IRES mediated translation. This will provide mechanistic insights into an essential component of the HCV life cycle that can be targeted therapeutically.



	CAP	*HCV IRES*
yeast	84160±3961	110±8
yeast+RRLBuff	18271±465	83±4
RRL	169707±11401	39905±1892
RRL+YeastBuff	14122±854	14606±866
Yeast+RRL	8666±235	97±6

Figure 4.4. The effect of mixing yeast and rabbit reticulocyte lysates on cap- and HCV IRES-dependent translation. Translation of cap (left) and HCV IRES (left)-dependent reporter RNAs (see Figure 4.3A) in yeast exract (dark blue), yeast extract diluted 1:1 with RRL extract buffer (light blue), RRL extract (red), RRL diluted 1:1 with yeast extract buffer (light red) and a 1:1 mixture of yeast and RRL extracts (green). Raw relative light units (RLU) are plotted and values are indicated on the right.

Chapter 5

Summary

5. Summary

Regulation of protein synthesis significantly influences the expression of most eukaryotic genes. The majority of characterized regulatory mechanisms target the rate-limiting stage of translation, initiation, where a translation-competent ribosomal complex is assembled and positioned on a messenger RNA. The large scaffolding protein eukaryotic initiation factor 4G (eIF4G) plays key roles in translation initiation, nucleating several interactions that are critical for ribosome recruitment. Diverse eukaryotes express multiple forms of eIF4G that make unique contributions to translation in mammals, flies, worms and plants. I extensively explored the function of the two eIF4G isoforms in *Saccharomyces cerevisiae* and observed a large functional overlap between the yeast eIF4G proteins. Both yeast isoforms have a similar capacity to support translation and growth under diverse environmental conditions and no messages expressed under standard laboratory conditions rely strongly on a specific isoform for their expression. Throughout these studies I observed that yeast are exquisitely sensitive to total levels of eIF4G. These findings suggest that, in addition to regulating the translation of specific transcripts, expressing eIF4G proteins from distinct genetic loci provides a robust means of controlling eIF4G levels, thus fortifying the translational apparatus and providing increased regulatory capacity. This work highlights the importance of distinguishing level from isoform-specific phenotypes, which has been overlooked in previous studies [163]. As part of this work, I developed a powerful tool, the isoform-specific (homogenic) strains, that deconvolutes level and isoform-specific contributions to eIF4G phenotypes and enables application of high-throughput genetic and phenotypic approaches to investigation of isoform-specific function.

Cellular perturbations, such as nutrient limitation, apoptosis and viral infection reduce the levels of full-length eIF4G, which is associated with inhibition of global translation initiation in diverse eukaryotes. I found that depletion of eIF4G modulates the translational efficiency of a narrow range (~5%) of yeast mRNAs, suggesting that eIF4G-limitation is also a mechanism of message-specific translational control. Despite an observed relationship between poly(A) tail length and changes in translational efficiency following eIF4G-limitation, directed *in vitro* studies suggest that eIF4G availability does not influence the ability of the poly(A) tail to enhance translation. Therefore, eIF4G limitation influences the translation of particular transcripts by a mechanism independent of poly(A) tail length. Consistent with a recent report [85], I observed a positive correlation between open reading frame length and changes in translational efficiency following eIF4G-limitation. Given the recently established connection between ORF length and stability of interactions between the 5'-cap and translation initiation factors [230], the influence of eIF4G-availability on the relationship between translational efficiency and ORF length of will interesting to explore mechanistically. These studies have created a robust data set to explore the mechanism underlying the modulation of translational efficiency by limiting eIF4G and, in doing so, further define the roles of eIF4G in translation initiation.

The Hepatitis C virus is a major public health problem, infecting more than 180 million people worldwide. Current treatments exhibit limited efficacy and are associated

with undesirable side effects. The HCV RNA utilizes an internal ribosome entry site to initiate the synthesis of proteins that are essential for viral replication and packaging. Therefore, this non-canonical mechanism of translation initiation represents a good drug target. In hopes of establishing a genetic system to elucidate mechanistic details of HCV translation, I characterized HCV IRES function in *Schizosaccharomyces pombe*. *S. pombe* expresses more mammalian-like translation factors than *Saccharomyces cerevisiae*, where robust HCV translation cannot be detected. I found that the HCV IRES, cannot efficiently initiate translation in *S. pombe in vivo* or *in vitro*, suggesting that additional mammalian factors are required for robust activity. These results, along with those from other labs (e.g. [232, 233]), suggest that HCV cannot efficiently initiate translation in fungi. These findings question the authenticity of modest HCV IRES-mediated translation recently observed in budding yeast, which could result from ribosomal read-through or the production of truncated RNAs by cryptic promoting or splicing events [234-236]. This work is a springboard for future studies aimed at identifying mammalian factors that enable robust HCV translation in fungi and open the possibility of applying a powerful genetic system to elucidation of the mechanism of HCV translation initiation, which will aid in drug design and comprehension of translational control mechanisms.

Appendix I

R Code

I. R Code

Code for various analyses is provided below. Raw data (other than microarray and that readily available from other published work) is provided in below. Additionally, input files (which are called directly by the code) are included along with the electronic copy of the text. Raw microarray data is available via ArrayExpress (http://www.ebi.ac.uk/arrayexpress/) accession numbers E-MEXP-2399, E-MEXP-2400, E-MEXP-2402 - see individual chapters for details).

Most the computational "legwork" is performed by a number of functions, which are called by "high level" code. In each section, the high level code is presented first to provide a conceptual framework and the detailed functions are then provided for more detailed reference below. To run this code you must first load the functions – either by coping directly into R or by "sourcing" a file containing them (which is provided: "functions.R") with the following code: `source(functions.R).`

I.1 *S. cerevisiae* eIF4G isoform pairwise alignment

The code presented here was used in the generation of Figure 2.2. Code used to make the scaled domain maps and the alignment heat map is below.

I.1.1 High Level Code

```
#read in the alignment score data
dir<- #USER MUST SPECIFY THE DIRECTORY CONTAINING ALIGNMENT SCORE DATA
source(paste(dir,"yeast4GalignmentQualityScoresForHeatMap.R", sep="/")) #make sure the
filename matches
scores<-alignmentScores

#function input
    windowSize<-3 #number of adjacent residues you want to average
    numCols<-50 #resolution of color spectrum (e.g. number of shades between high and low
color)
    cols<-c("dodgerblue4","gold") #low and high color, respectively
    tickDist<-50 #distance between tics on the axis label
    numResidues<-1000 #scale for the plot; how many residues to show

pdf(file=paste(dir,"figure1b.pdf",sep="/"))
par(mfcol=c(4,2))

#draw the domain maps
draw4GdomainMaps(tickDist, numResidues=numResidues) #see function below for details

#draw the alignment domain and heat map
alignmentHeatMap(scores=scores, numCols=numCols, windowSize=windowSize, cols=cols,
tickDist=tickDist, numResidues=numResidues)

dev.off()
```

I.1.2 Functions

I.1.2.1 draw4GdomainMaps

```
draw4GdomainMaps<-function(tickDist, numResidues){
    #4G1
    tif4631<-matrix(data=c(0), nrow=952, ncol=1) #background
    tif4631[c(1:82),]<-2 #RNA1
    tif4631[188:299,]<-4 #pab
    tif4631[441:490,]<-6 #4E
    tif4631[452:458,]<-8 #4E;hpo motif
    tif4631[492:539,]<-10 #RNA2/RS
    tif4631[542:883,]<-12 #4a
```

```
    tif4631[c(883:952),]<-14 #RNA3/RS

    image(  x=c(0:length(tif4631)), y=0.1, z= tif4631,
                    col = c("gray35","burlywood3","darkorange2", "chartreuse4", "darkgreen",
                           "rosybrown1","royalblue", "rosybrown1"),
                    xlab="Residue Number", ylab="", axes="F", breaks=c(0,1,3,5,7,9,11,13,15),
                    xlim=c(0,numResidues), main="TIF4631")
    axis(side=1, at=seq(from=0, to=numResidues, by= tickDist))

    #4G2
    tif4632<-matrix(data=c(0), nrow=914, ncol=1) #background
    tif4632[201:315,]<-2 #pab
    tif4632[400:515,]<-4 #4E
    tif4632[423:429,]<-6 #4E;hpo motif
    tif4632[439:914,]<-8 #4A


    image(  x=c(0:length(tif4632)), y=0, z= tif4632,
                    col = c("gray72","darkorange2","chartreuse4", "darkgreen", "royalblue"),
                    xlab="Residue Number", ylab="", axes="F", breaks=c(0,1,3,5,7,9),
                    xlim=c(0,numResidues), main="TIF4632")
    axis(side=1, at=seq(from=0, to=numResidues, by= tickDist))}
```

1.1.2.2 alignmentHeatMap

```
alignmentHeatMap<-function(scores,numCols, windowSize, cols, tickDist, numResidues){
    library(gplots)

    #first make the non-sliding winow plot; each residue is colored individually
    image(  x=c(0:length(scores)), y=0, z=scores,
                    col = colorpanel(n=numCols,low=cols[1],high=cols[2]),
                    xlab="Residue Number", ylab="", axes="F",
                    xlim=c(0,numResidues), main="Aligment Scores HeatMap Single Nucleotide
Resolution")
    axis(side=1, at=seq(from=0, to=numResidues, by=tickDist))

    #now do the sliding window average
    #a matrix of NAs to be filled in with the averages
    windowScores<-matrix(nrow=length(scores),ncol=1)

    #parameters of the sliding window
    #if it is even sized the extra value will be to the left;
            #e.g. the average for entry 4 will be of 2:5
    leftWindowSize<-windowSize%/%2
    rightWindowSize<-(windowSize-leftWindowSize)
    start<-1+windowSize%/%2
    end<-(length(scores)-rightWindowSize)

    #for the beginning of the list;
    #all entries within 1/2 the window size of the start will
            #get the same average - that within the first window
    windowScores[1:start]<-mean(c(scores[1:windowSize]))

    #for the middle
    #push the window along one entry at a time and calculate the average of the entries that
it spans
    for(i in start:end){
            windowScores[i,]<-mean(c(scores[(i-leftWindowSize):(i+rightWindowSize-1)]))}

    #for the end
    #all entries within 1/2 the window size of the end will
```

```
        #all get the same average - that covered by the window placed
        #all the way at then end
    windowScores[(length(scores)-rightWindowSize+1):length(scores)]<-
        mean(c(scores[(length(scores)-windowSize+1):length(scores)]))
    image(  x=c(0:length(windowScores)), y=0, z=windowScores,
            col = colorpanel(n=numCols,low=cols[1],high=cols[2]),
            xlab="Residue Number", ylab="", axes="F",
            xlim=c(0,numResidues), main="Aligment Scores HeatMap Sliding Window")
    axis(side=1, at=seq(from=0, to=numResidues, by=tickDist))

    #to make a key for what the colors mean
    colRange<-matrix(     data=c(seq(from=range(scores)[1],to=range(scores)[2],
                          by=(range(scores)[2]-
range(scores)[1])/numCols)),nrow=numCols+1,ncol=1)

    image(x=colRange, z=colRange,col=colorpanel(n=numCols,low=cols[1],high=cols[2]),axes="F",
            ylab="Color Key", xlab="BLOSUM62 Scores")
    axis(side=1, at=round(seq(from=0, to=range(scores)[2], by=(range(scores)[2]/5)),
digits=2))

    #to show where domains (use the 4G1 coordinates and then map them to the gapped 4G1
sequence
    #(which has the same spacig as the consensus) and then highlight these residues) fall on
the consensus
    consensus<-matrix(data=c(0), nrow=978, ncol=1) #background
    consensus[206:319,]<-2 #pab
    consensus[564:905,]<-4 #4a
    consensus[c(1:88),]<-6 #RNA1
    consensus[c(514:561,905:978),]<-8 #RNA2/RS
    consensus[463:512,]<-10 #4E
    consensus[474:480,]<-12 #4E;hpo motif

    image(  x=c(0:length(consensus)), y=0.1, z= consensus,
            col = c("white","darkorange2","royalblue", "burlywood3",
                    "rosybrown1", "chartreuse4", "darkgreen"),
            xlab="Residue Number", ylab="", axes="F", breaks=c(0,1,3,5,7,9,11,13),
            xlim=c(0,numResidues), main="Consensus Domain Locations")
    axis(side=1, at=seq(from=0, to=numResidues, by= tickDist))

    #for the black line that will run the length of the consensus
    image(  x=c(0:length(consensus)), y=0.1, z= consensus,
            col = c("black"), xlab="", ylab="", axes="F", xlim=c(0,numResidues),
            main="Black Line for Consensus")

    #for the gaps in the consensus; these represent spots where gaps had to be introduced in
either
    #sequence during the aligment (i.e. where sequence in the consensus come from a single
isoform )
    gaps<-matrix(data=NA, nrow=978, ncol=1) #background
    gaps[c(82:87,99:104,140:144,175,209:210,372:373,909:911,923,249:262,291:292,
                    320:333,350:360,412:419,523:528,576:579,956:957),]<-0 #pab
    image(  x=c(0:length(gaps)), y=0.1, z= gaps,
            col = c("white"), xlab="", ylab="", axes="F", xlim=c(0,numResidues),
            main="For Gaps in Consensus")
    axis(side=1, at=seq(from=0, to=numResidues, by= tickDist))}
```

I.2 Microarray Analysis
I.2.1 High Level Code

I.2.1.1 Ribosome Occupancy

```
#Ribosome occupancy
#input
targetsFName<-"ribosomeOccupancyTargets.txt" #provided as supplemental material
outputFName<-"ribosomeOccupancy"
arraysToDwnWt<-NULL
chipType<-"y01"
etype<-"PvT"
pval<-0.05
makeListFiles<-"f"
design <- cbind(
WT_1  = c(1,0,0,-1,0,0,0,0,0,0,0,0),
WT_2  = c(0,0,0,0,0,0,1,0,0,-1,0,0),
e4G1_1= c(0,1,0,0,-1,0,0,0,0,0,0,0),
e4G1_2= c(0,0,0,0,0,0,0,1,0,0,-1,0),
e4G2_1= c(0,0,1,0,0,-1,0,0,0,0,0,0),
e4G2_2= c(0,0,0,0,0,0,0,0,1,0,0,-1))

contrast <- cbind(
        WT = c(1,1,0,0,0,0),
        e4G1 = c(0,0,1,1,0,0),
        e4G2 = c(0,0,0,0,1,1),
        e4G1mWT = c(-1,-1,1,1,0,0),
        e4G2mWT = c(-1,-1,0,0,1,1),
        e4G1me4G2 = c(0,0,1,1,-1,-1),
        e4G2me4G1 = c(0,0,-1,-1,1,1))

doAnalysis(dir=dir,targetsFName= targetsFName,outputFName= outputFName,arraysToDwnWt=
arraysToDwnWt,
        design= design,contrast= contrast,etype= etype, chipType= chipType, pval=pval,
makeListFiles="f")
```

I.2.1.2 Transcript Abundance

```
#Transcript Abundance
#input
targetsFName<-"transcriptAbundanceTargets.txt" #provided as supplemental material
outputFName<-"transcriptAbundance"
arraysToDwnWt<-NULL
chipType<-"y01"
etype<-"TvT"
pval<-0.05
makeListFiles<-"f"
design <- cbind(
WT_1  = c(1,0,0,-1,0,0,0,0,0,0,0,0),
WT_2  = c(0,0,0,0,0,0,1,0,0,-1,0,0),
e4G1_1= c(0,1,0,0,-1,0,0,0,0,0,0,0),
e4G1_2= c(0,0,0,0,0,0,0,1,0,0,-1,0),
e4G2_1= c(0,0,1,0,0,-1,0,0,0,0,0,0),
e4G2_2= c(0,0,0,0,0,0,0,0,1,0,0,-1))

contrast <- cbind(
        WT = c(1,1,0,0,0,0),
        e4G1 = c(0,0,1,1,0,0),
        e4G2 = c(0,0,0,0,1,1),
        e4G1mWT = c(-1,-1,1,1,0,0),
        e4G2mWT = c(-1,-1,0,0,1,1),
```

```
        e4G1me4G2 = c(0,0,1,1,-1,-1),
        e4G2me4G1 = c(0,0,-1,-1,1,1))

doAnalysis(dir=dir,targetsFName= targetsFName,outputFName= outputFName,arraysToDwnWt=
arraysToDwnWt,
        design= design,contrast= contrast,etype= etype, chipType= chipType, pval=pval,
makeListFiles="f")
```

I.2.1.3 Polysome Abundance

```
#Polysome Abudance
#input
targetsFName<-"polysomeAbundanceTargets.txt"
outputFName<-"polysomeAbundance"
arraysToDwnWt<-NULL
chipType<-"y01"
etype<-"PvP"
pval<-0.05
makeListFiles<-"t"
design <- cbind(
WT_1  = c(1,0,0,-1,0,0,0,0,0,0,0,0),
WT_2  = c(0,0,0,0,0,0,1,0,0,-1,0,0),
e4G1_1= c(0,1,0,0,-1,0,0,0,0,0,0,0),
e4G1_2= c(0,0,0,0,0,0,0,1,0,0,-1,0),
e4G2_1= c(0,0,1,0,0,-1,0,0,0,0,0,0),
e4G2_2= c(0,0,0,0,0,0,0,0,1,0,0,-1))

contrast <- cbind(
        WT = c(1,1,0,0,0,0),
        e4G1 = c(0,0,1,1,0,0),
        e4G2 = c(0,0,0,0,1,1),
        e4G1mWT = c(-1,-1,1,1,0,0),
        e4G2mWT = c(-1,-1,0,0,1,1),
        e4G1me4G2 = c(0,0,1,1,-1,-1),
        e4G2me4G1 = c(0,0,-1,-1,1,1))

doAnalysis(dir=dir,targetsFName= targetsFName,outputFName= outputFName,arraysToDwnWt=
arraysToDwnWt,
        design= design,contrast= contrast,etype= etype, chipType= chipType, pval=pval,
makeListFiles="f")
```

I.2.2 Functions

I.2.2.1 doAnalysis

```
################
#doAnalysis
################
#performs a statistical analysis of microarrays using a number of
#additional functions below

doAnalysis<-function(dir, targetsFName, outputFName, arraysToDwnWt,
               design, contrast, etype, chipType, pval, makeListFiles){

##########################################
##########################################
#load required libraries and homemade functions
        library(limma)
        library(arrayQuality)
        library(ABarray)
        setwd(dir)
        source("E12qualityPlotsFunction.R")
```

```
	source("E12Functions.R")

########################################
########################################
#1. READ IN DATA AND MANIPULATE THE RG OBJECT

	#load file with chip IDs
		targets<-readTargets(file= targetsFName)
		RG<-read.maimages(targets$FileName, source="genepix.median",
			wt.fun=wtflags(0.01,0))

	#swap the "ID" and "Name" column names in yOligo arrays to facilitate
	#downstream analysis
		if(chipType=="yOligo"){
			colnames(RG$genes)<-c("Block","Row","Column","Name","ID")}

	#Define what samples you have
	#NOTE: ASSUMES A PARTICULAR EXPERIMENTAL DESIGN
		sampNames<-c("WT-1a","e4G1-1a", "e4G2-1a","WT-1b","e4G1-1b", "e4G2-1b",
					"WT-2a","e4G1-2a", "e4G2-2a","WT-2b","e4G1-2b", "e4G2-
2b")

	#create an object with sensible array name descriptions
	#NOTE: the sampNames vector is recycled if there are more than
	#12 arrays in the targets file
		arrayNames<-paste(sampNames,"(",targets[,"SlideNumber"],")",sep="")

	#use arrayNames object to give more sensible names to RG object
		colnames(RG)<-arrayNames
		RG$targets[,1]<-arrayNames

	#give names/ID's from new gal file
		if(chipType=="y01"){
			newGal<-read.csv("ReorderedgodlistGalFile060309.csv",
				stringsAsFactors=FALSE)
			RG$genes$ID<-newGal$ID
			RG$genes$Name<-newGal$Name}

########################################
########################################
#2. MAKE QUALITY PLOTS

	#get indicies for which arrays involve which samples
		groupList <- list(WT=c(sort(c(grep("^WT_.*",targets$Sample)))),
					e4G1=c(sort(c(grep("^.4G1_.*",targets$Sample)))),
					e4G2=c(sort(c(grep("^.4G2_.*",targets$Sample)))))
	#make the series of plots using qualityPlots function (see below for details)
		print("making qualityPlots")
		pdf(paste(outputFName,"qualityPlots.pdf",sep=""))
		qualityPlots(RG=RG,doBkg=TRUE,arrayNames=arrayNames,groupList=groupList,
			etype=etype,intensityCutoff=8,chipType=chipType,dir)
		dev.off()
		print("done")

########################################
########################################
#3. DOWNWEIGHT SUSPECT ARRAYS AND CONTROL SPOTS

	#check to see if there are any arrays to downweight
		if(length(arraysToDwnWt)>1){
```

```
                        for(i in 1:length(arraysToDwnWt)){
                                RG$weights[,arrayNames[arraysToDwnWt[i]]]<-0.01}}
                else{
                        if(length(arraysToDwnWt)==1){
                        RG$weights[,arrayNames[arraysToDwnWt[1]]]<-0.01}}

        #use dwnWTCntrls function (see below for details)
        #to downweight suspect arrays
                print("downweighting control spots")
                        RG<-dwnWtCntrls(RG,chipType=chipType,dir=dir)
                print("done")
                        save(RG,file=paste(outputFName,"RG.Rdata",sep=""))

##########################################
##########################################
#4. PRE-PROCESS

        #perform pre-processing using preProcess function (see below for details)
                MAlist<-preProcess(RG)
                RGlist<-lapply(MAlist, RG.MA)
                save(MAlist, file=paste(outputFName,"MAlist.Rdata",sep=""))

        #load design matrix
                design<-design
                rownames(design)<-arrayNames

        #adjust design to downweight suspect arrays
                if(length(arraysToDwnWt)>1){
                for(i in 1:length(arraysToDwnWt)){
                design[arrayNames[arraysToDwnWt[i]],]<-0}}
                else{if(length(arraysToDwnWt)==1){
                        design[arrayNames[arraysToDwnWt[i]],]<-0}}

        #load contrast matrix
                contrast<-contrast
                rownames(contrast)<-colnames(design)

##########################################
##########################################
#5. INCORPORATE WITHIN-ARRAY DUPLICATE SPOTS

        #put within array dups next to each other in list using
        #twoSpot function (see below for details)
                print("processing twoSpot")
                twoSpotMAlist<-lapply(c("Raw","Loess","AquantScale"), function(i){
                                twoSpot(MAlist[[i]])})
                names(twoSpotMAlist)<-c("Raw","Loess","AquantScale")
                print("     done")

        #calculate the replicate correlation
                print("processing corFit")
                corFitMAtwoSpotlist<-lapply(twoSpotMAlist, corFit, ndups=2,
                        spacing=1, block=NULL, design=design)
                print("     done")

        #add the correlation to the MAlist
                for(i in 1:length(twoSpotMAlist)){
                        twoSpotMAlist[[i]]$cor<-corFitMAtwoSpotlist[[i]]$consensus.correlation}

        save(corFitMAtwoSpotlist, file=paste(outputFName,
```

```
                                        "corFitTwoSpotMAlist.Rdata",sep=""))
        save(twoSpotMAlist, file=paste(outputFName,"twoSpotMAlist.Rdata",sep=""))

########################################
########################################
#6. FIT LINEAR MODEL

        #perform lmfit, fit.contrasts, and calculate eBayes stats (coef, pvalue, etc.)
        #for each gene for each contrast using linearModel function (see below for details)
                print("fitting linear model...")
                fitContrastTwoSpotMAlist<-lapply(twoSpotMAlist, linearModel, cor="t", ndups=2,
                        spacing=1, block=NULL, design=design, contrast=contrast)

                save(fitContrastTwoSpotMAlist, file=paste(outputFName,
                                        "fitContrastTwoSpotMAlist.Rdata",sep=""))
                print("     done")

########################################
########################################
#6. MAKE TABLE OBJECTS SUMMARIZING

        #correct for multiple testing and look at everything with a
        #significant pval (<= 0.05)

        #create object with just gene names (suitable for plugging into GO analysis)
        #with GOList function (see below for details)
                print("topTable objects...")
                topTwoSpotMAgoList<-lapply(fitContrastTwoSpotMAlist, GOList, pval=pval,
                        file="f" , ret="t", chipType=chipType, fname=outputFName,dir=dir)

        #create object with all the topTable data for the significant genes
        #usiing topTableList function (see below for details)
                topTwoSpotMAtableList<-lapply(fitContrastTwoSpotMAlist, topTableList, pval=1,
                        ret="t", chipType=chipType, fname=outputFName, trim="f",
split="f",dir=dir,
                        addAnn=makeListFiles)

        save(topTwoSpotMAgoList, file=paste(outputFName,
                                "topTwoSpotMAgoList.Rdata",sep=""))
        save(topTwoSpotMAtableList, file=paste(outputFName,
                                "topTwoSpotMAtableList.Rdata",sep=""))
        print("     done")


########################################
########################################
#7. OUTPUT LISTS

        #export list amenable to GO analysis (just gene names)
                if(makeListFiles=="t"){
                print("topTable files...")
                #files for plugging into the GO finder
                sapply(c(1,3,6), function(i){
                        sapply(1:length(topTwoSpotMAgoList[[i]]), function(j){
                                write(as.character(unlist(topTwoSpotMAgoList[[i]][j])),
                                                file=paste(outputFName,
names(topTwoSpotMAgoList)[i],
                                                names(topTwoSpotMAgoList[[i]])[j], ".txt",sep=""),
                                                sep=" \n")})})
```

```
	#files with more extensive data; include all of the data;
	#not just the significant stuff
		sapply(c(1,3,6), function(i){
			write.table(topTwoSpotMAtableList[[i]],
					file=paste(outputFName, names(topTwoSpotMAgoList)[i],
					".tab",sep=""))})

		print("     done")}


##########################################
##########################################
#7. SUMMARY PLOTS

	comparisons<-names(topTwoSpotMAgoList)
	pdf(paste(dir,outputFName,"sigFigs.pdf",sep=""))

	#barplot of the number of genes in 4 categories (i4G1up, i4G1dn, i4G2up, i4G2dn);
		par(mfrow=c(2,2),family="mono")
		sigGenesPlot(topTwoSpotMAgoList, toPlot=c(1:3))

	#venn diagrams to look at overlap between the processing steps
		vennDiagrams(topTwoSpotMAgoList, comparisons)

	#MA-plot of contrasts
		MAplot(fitContrastTwoSpotMAlist[["AquantScale"]],coef="e4G1mWT",
			pval=pval,intCO=8,lfcCO=0.58, mName="Loess e4G1-WT")
		MAplot(fitContrastTwoSpotMAlist[["AquantScale"]],coef="e4G2mWT",
			pval=pval,intCO=8,lfcCO=0.58, mName="Loess e4G2-WT")
		MAplot(fitContrastTwoSpotMAlist[["AquantScale"]],coef="e4G1me4G2",
			pval=pval,intCO=8,lfcCO=0.58, mName="Loess e4G1-e4G2")

	#comparison scatterplots of contrasts
		comPlot(fitContrastTwoSpotMAlist[["AquantScale"]],coef1="e4G1",coef2="WT",
			pval=pval,intCO=8,lfcCO=0.58, mName="Loess e4G1-WT")
		comPlot(fitContrastTwoSpotMAlist[["AquantScale"]],coef1="e4G2",coef2="WT",
		pval=pval,intCO=8,lfcCO=0.58, mName="Loess e4G2-WT")
		comPlot(fitContrastTwoSpotMAlist[["AquantScale"]],coef1="e4G1",coef2="e4G2",
			pval=pval,intCO=8,lfcCO=0.58, mName="Loess e4G1-e4G2")
dev.off()
print(paste(outputFName, "done", sep=""))
warnings()}
```

I.2.2.2 dwnWtCntrls

```
################
#dwnWtCntrls
################
#downweights control spots that were included on the array.
#these can have cross hybridization and since i didn't
#include any control RNA in the experiments are not
#included in the analysis (so they wont have any influence
#on the statistics)

dwnWtCntrls<-function(RG,chipType,dir){
	setwd(dir)

	if(chipType=="y01"){
		#made a .csv file that has all the types lined up with the .gpr list;
		#made this in excel and have checked it
		UCSFGPRtype<-read.csv("UCSFGPRtype.csv", colClasses="character")
```

```
        RG$genes$Type<-UCSFGPRtype[,2]
        RG$weights<-modifyWeights(RG$weights, RG$genes$Type,
                        c("CAS","CDP","CMM","CMO","EMPTY","TAG","0"),c(0.01))
        return(RG)}

    if(chipType=="yOligo"){
    stanGlist<-read.csv("Stanford_Godlist_6_19_07_2.csv",
        colClasses="character")
    stanGlist$Index<-as.numeric(stanGlist$Index)

    stanGlistIndex<-read.csv("yOligo_StanGlist_Index.csv",
        colClasses="character")
    stanGlistIndex$CorStanGlistIndex<-as.numeric(stanGlistIndex$CorStanGlistIndex)
    #throws a warning about NA introduction; this is for the small
    #number (5) of features that are on the gpr file and not on the
    #godlist, so they don't have a corresponding index

    RG$genes$StanGlistIndex<-stanGlistIndex$CorStanGlistIndex
    #indicies created by hand
    RG$genes$Type<-lapply(
    RG$genes$StanGlistIndex,             #for each Godlist index assigned to that probe;
    function(i){stanGlist$TYPE[which(stanGlist$Index==i)]})
        #pull the TYPE from the Godlist
        RG$genes$Type<-as.character(RG$genes$Type)

    RG$weights<-modifyWeights(RG$weights, RG$genes$Type,
        c("CONTROL","MARKER","EMPTY"),c(0.01))

    return(RG)}}
```

I.2.2.3 preProcess

```
################
#preProcess
################
#preProcess an RG object.  will produce a list with
                    #1=raw
                    #2=normexp BG correction (offset=50)
                    #3=normexp BG correction (offset=50)+loess
                    #4=normexp BG correction (offset=50)+loess+aquantile
                    #5=normexp BG correction (offset=50)+loess+scale
                    #6=normexp BG correction (offset=50)+loess+aquantile+scale

preProcess<-function(x){
    objectMAList<-list(Raw=0,BGcor=0,Loess=0,Aquantile=0,Scale=0,
        AquantScale=0)

    #raw MA objects
        objectMAList[[1]]<-MA.RG(x, bc.method="none")

    #Background Correct
        objectMAList[[2]]<-MA.RG(x, bc.method="normexp", offset=50)

    #Normalize Within Arrays
        objectMAList[[3]]<-normalizeWithinArrays(
                                                    objectMAList[[2]],
                                                    method="printtiploess",
                                                    bc.method="none",
                                                    weights=x$weights)

    #Normalize Between Arrays
```

```
	#Aquantile
		objectMAList[[4]]<-normalizeBetweenArrays(objectMAList[[3]],
			method="Aquantile")
	#Scale
	objectMAList[[5]]<-normalizeBetweenArrays(objectMAList[[3]],
		method="scale")
	#Aquantile-Scale
	objectMAList[[6]]<-normalizeBetweenArrays(objectMAList[[4]],
		method="scale")

	return(objectMAList)}
```

I.2.2.4 twoSpot

```
################
#twoSpot
################
#will produce an alphabetical list of probes with 2 reps of each listed
#one after another

twoSpot<-function(x){

	#order alphabetically
		x<- x[order(x$genes$ID),]

	#get all the unique names
		uniq<-unique(x$genes$ID)
		#8599 entries for yOligo arrays; 8552 for y01
	#initialize a column to keep track of the rep number of each probe
	#grab all indexes of each uniq entry from full list;
	#length of y = # of occurences of uniq entry
	#use the indicies to fill in the replicate number
	#(1:length(y) spits out 1,2,3...) of that probe
		x$genes$spotrep[1:length(x$genes$ID)] <- 0
		for (i in 1:length(uniq)) {
			y <- which(x$genes$ID == uniq[i]);
			x$genes$spotrep[y] <- 1:length(y)}

	#look for spots with just 1 occurence;
	#subtract those with occurence = 2 from occurence = 1
		loners<-setdiff(x$genes$ID[x$genes$spotrep == 1],
					x$genes$ID[x$genes$spotrep == 2])
					#10 with yOligo arrays (all TIL); 0 with y01
		if(length(loners >0)){
			for(i in 1:length(loners)){
			#take the rows that don't match the loners
			x<-x[which(x$genes$ID != loners[i]),]}}

	#new MA object that includes only the data for the first 2 or less
	#occurrences of a probes
	#17178 rows with yOligo; 17104 with y01
	twoSpotMA <- x[x$genes$spotrep <= 2 &
					x$genes$spotrep != 0, ]

	# now your spacing=1 and ndups=2
	return(twoSpotMA)}
```

I.2.2.5 corFit

```
################
#corFit
################
#determines replicate correlation for within array reps

corFit<-function(x, ndups, spacing, block, design){

        library(statmod)
        corFit<-duplicateCorrelation(x,design=design, ndups=ndups,
                        spacing=spacing, block=block, weights=x$weights)
        return(corFit)}
```

I.2.2.6 linearModel

```
################
#linearModel
################
#performs lmFit and contrast.fit functions on an MA object (x) with a
#given design and contrast matrix and takes within array duplicate spots
#into account (requires the product of the twoSpot and corFit functions above)

linearModel<-function(x,cor,ndups,spacing,block,design,contrast){
        if(cor=="t"){
                fit<-lmFit(x, design, ndups, spacing, block, correlation=x$cor,
                        weights=x$weights)
                fit.contrast<-contrasts.fit(fit,contrast)
                fit.contrast<-eBayes(fit.contrast)}
        else{
                fit<-lmFit(x, design, ndups, spacing, block, weights=x$weights)
                fit.contrast<-contrasts.fit(fit,contrast)
                fit.contrast<-eBayes(fit.contrast)}

        return(fit.contrast)}
```

I.2.2.7 GOList

```
################
#GOList
################
#determines significant genes for different strain comparisons,
#writes out a text file (and returns data if desired) with
#SGD GOfinder friendly names (will homogenize chips from different
#print runs if needed)

GOList<-function(x, pval, file, ret, chipType, fname, dir){
        if(length(x)>0){
        #get the lists; pvalue filter here is based on adjusted pvalue
                sigGenes<-list(
                e4G1mWT=topTable(x, coef="e4G1mWT", n=Inf, genelist=x$genes$ID,
                                                adjust.method="BH", sort.by="none", p.value=pval,
lfc=0),
                e4G1me4G2=topTable(x, coef="e4G1me4G2", n=Inf, genelist=x$genes$ID,
                                                adjust.method="BH", sort.by="none", p.value=pval,
lfc=0),
                e4G2mWT=topTable(x, coef="e4G2mWT", n=Inf, genelist=x$genes$ID,
                                                adjust.method="BH", sort.by="none", p.value=pval,
lfc=0),
                e4G2me4G1=topTable(x, coef="e4G2me4G1", n=Inf, genelist=x$genes$ID,
                                                adjust.method="BH", sort.by="none", p.value=pval,
lfc=0))
```

```
        #convert yOligo names
        if(chipType=="yOligo"){
                setwd(dir)
                nameConverter<-read.csv("yOligoNameConverter.csv", colClasses="character")
                converter<-function(i,x){
                        if(dim(i)>0){
                        for(j in 1:dim(i)[1]){
                                if(length(which(nameConverter$StanName==i$ID[j]))>0){
                                i$ID[j]<-nameConverter$ID[which(nameConverter$StanName==i$ID[j])]
                                }
                                #clean up the names take only the text preceeding the first
                                #"_" for the ORF features
                                if(length(sub("(_.*)","",i$ID[j]))>0 & x$genes$Type[j]=="ORF"){
                                i$ID[j]<-sub("(_.*)","",i$ID[j]) }}
                return(i)}else(return(i))}
                sigGenes<-lapply(sigGenes,converter,x)}

        #write out text file with names
        if(file=="t"){
                setwd(dir)
                sapply(1:length(sigGenes),function(i){
                        write(as.character(sigGenes[[i]]$ID[which(sigGenes[[i]]$logFC>0)]),
                                file=paste(fname,names(sigGenes)[i],"up",pval,".txt", sep=""),
sep=" \n")
                        write(as.character(sigGenes[[i]]$ID[which(sigGenes[[i]]$logFC<0)]),
                                file=paste(fname,names(sigGenes)[i],"dn",pval,".txt", sep=""),
sep=" \n")})}
        #return a datafile if desired
        if(ret=="t"){
                dataList<-lapply(sigGenes,function(i){
                        return(list(up=as.character(i$ID[which(i$logFC>0)]),
                                        down=as.character(i$ID[which(i$logFC<0)])))  })
                dataList<-unlist(dataList,recursive=FALSE)
                return(dataList)}}}
```

I.2.2.8 topTableList

```
################
#topTableList
################
#creates an excel-friendly file for each strain comparison made that summarizes
#all of the stats

#input: contrastFit object
#arguments:
        #x=fit object
        #pval = adj p.value cutoff; only data below this cutoff will be returned/printed
        #file = t or f; do you want a file to be produced?
        #ret = t or f; do you want an object returned from the function
        #fname = filename for file production
        #trim = do you want the yOligo names trimmed?
        #split = do you want the table split into separate ones for up/down for each coef?

topTableList<-function(x, pval, ret, chipType, fname, trim, split, dir, addAnn){
        if(length(x)>0){

        #compare strain as indicated in contrast matrix
                sigGenes<-list(
                e4G1mWT=topTable(x, coef="e4G1mWT", n=Inf, genelist=x$genes$ID,
```

```
                                        adjust.method="BH", sort.by="none", p.value=pval,
lfc=0),
                e4G1me4G2=topTable(x, coef="e4G1me4G2", n=Inf, genelist=x$genes$ID,
                                        adjust.method="BH", sort.by="none", p.value=pval,
lfc=0),
                e4G2mWT=topTable(x, coef="e4G2mWT", n=Inf, genelist=x$genes$ID,
                                        adjust.method="BH", sort.by="none", p.value=pval,
lfc=0),
                e4G2me4G1=topTable(x, coef="e4G2me4G1", n=Inf, genelist=x$genes$ID,
                                        adjust.method="BH", sort.by="none", p.value=pval,
lfc=0))

        #add the stanford information to the sigGenes objects
                sigGenes<-lapply(sigGenes, function(i){
                        if(dim(i)[1]>0){
                        i$StanID<-x$genes$ID
                        i$StanIndex<-x$genes$Name
                        return(i)}})

        #convert yOligo names
                if(chipType=="yOligo"){
                setwd(dir)
                nameConverter<-read.csv("yOligoNameConverter.csv", colClasses="character")
                converter<-function(i){
                        if(dim(i)[1]>0){
                        for(j in 1:dim(i)[1]){
                                if(length(which(nameConverter$StanName==i$genes$ID[j]))>0){
                                i$genes$ID[j]<-nameConverter$ID[which(
                                        nameConverter$StanName==i$genes$ID[j])]}
                                if(trim =="t"){
                                        if(length(substring(i$genes$ID[j],1,1))>0){
                                        if(substring(i$genes$ID[j],1,1)=="Y"|

        substring(i$genes$ID[j],1,1)=="y"){
                                                if(length(sub("(_.*)","",i$genes$ID[j]))>0){
                                                i$genes$ID[j]<-sub("(_.*)","",i$genes$ID[j])}}}}}
                        return(i)}}
                sigGenes<-lapply(sigGenes,converter)}

        #add annotations to table
                chromosomeLengths<-c(230208,813178,316617,1531918,
                                                        576869,270148,1090947,562643,
                                                        439885,745745,666454,1078175,
                                                        924429,784333,1091289,948062,85779)

                setwd(dir)
                y01Annotator<-read.csv(file="y01Annotator.csv", colClasses="character")
                addAnnotations<-function(x){
                                if(length(x)>0){
                                commonNames<-sapply(1:dim(x)[1],function(j){
                                        if(length(which(y01Annotator$ID==x$ID[j]))==1){
                                                return(as.character(y01Annotator$standardName[which(
                                                        y01Annotator$ID==x$ID[j])]))}
                                        if(length(which(y01Annotator$ID==x$ID[j]))>1){
                                                return(as.character(y01Annotator$standardName[which(
                                                        y01Annotator$ID==x$ID[j])[1]]))}
                                        if(length(which(y01Annotator$ID==x$ID[j]))<1){
                                                return(x$ID[j])}})
                                descriptions<-as.character(sapply(1:dim(x)[1],function(i){
                                        if(length(which(y01Annotator$ID==x$ID[i]))==1){
```

```
                                        return(as.character(y01Annotator$description[which(
                                                y01Annotator$ID==x$ID[i])]))}
                                if(length(which(y01Annotator$ID==x$ID[i]))>1){
                                        return(as.character(y01Annotator$description[which(
                                                y01Annotator$ID==x$ID[i])[1]]))}
                                if(length(which(y01Annotator$ID==x$ID[i]))<1){
                                        return(NA)}}))
                        chromosomes<-as.character(sapply(1:dim(x)[1],function(i){
                                if(length(which(y01Annotator$ID==x$ID[i]))==1){
                                        if(as.character(y01Annotator$chromosome[which(
                                                y01Annotator$ID==x$ID[i])])=="#N/A"){
                                                return(NA)}
                                        else{

return(as.character(y01Annotator$chromosome[which(
                                                        y01Annotator$ID==x$ID[i])]))}}
                                if(length(which(y01Annotator$ID==x$ID[i]))>1){
                                        if(as.character(y01Annotator$chromosome[which(
                                                y01Annotator$ID==x$ID[i])[1]])=="#N/A"){
                                                return(NA)}
                                        else{

return(as.character(y01Annotator$chromosome[which(
                                                        y01Annotator$ID==x$ID[i])[1]]))}}
                                if(length(which(y01Annotator$ID==x$ID[i]))<1){
                                        return(NA)}}))
                        minCoordinates<-as.numeric(sapply(1:dim(x)[1],function(i){
                                if(length(which(y01Annotator$ID==x$ID[i]))==1){
                                        if(as.character(y01Annotator$minCoordinate[which(
                                                y01Annotator$ID==x$ID[i])])=="#N/A"){
                                                return(NA)}
                                        else{
                                        return(as.numeric(y01Annotator$minCoordinate[which(
                                                y01Annotator$ID==x$ID[i])]))}}
                                if(length(which(y01Annotator$ID==x$ID[i]))>1){
                                        if(as.character(y01Annotator$minCoordinate[which(
                                                y01Annotator$ID==x$ID[i])[1]])=="#N/A"){
                                                return(NA)}
                                        else{

return(as.numeric(y01Annotator$minCoordinate[which(
                                                        y01Annotator$ID==x$ID[i])[1]]))}}
                                if(length(which(y01Annotator$ID==x$ID[i]))<1){
                                        return(NA)}}))
                        normChromLoc<-as.numeric(sapply(1:dim(x)[1],function(i){

minCoordinates[i]/chromosomeLengths[as.numeric(chromosomes[i])]}))

                        chromosomes[which(chromosomes==17)]<-"M"

                        x$standardName<-commonNames
                        x$descriptions<-descriptions
                        x$chromosomes<-chromosomes
                        x$minCoordinates<-minCoordinates
                        x$normChromLoc<-normChromLoc
                        return(x)}}

        if(addAnn=="t"){
        sigGenes<-lapply(sigGenes, addAnnotations)}
```

```
		if(ret=="t"){
			if(split=="t"){
				dataList<-lapply(sigGenes,function(i){
				up<-i[which(i$logFC>0),]
				down<-i[which(i$logFC<0),]
				return(list(up=up,down=down))})
			dataList<-unlist(dataList,recursive=FALSE)
			return(dataList)}
			if(split=="f"){
				return(sigGenes)}}}}
```

I.2.2.9 writeTopTable

```
################
#writeTopTable
################
#produce excel friendly files from topTable objects

writeTopTable<-function(topTable,dir,split,fname,toWrite){
		setwd(dir)
		sapply(toWrite,function(j){
			if(split=="t"){
			sapply(1:length(topTable[[j]]),function(i){
				write.table(topTable[[j]][[i]][which(topTable[[j]][[i]]$logFC>0),],
						file=paste(fname,names(topTable[[j]])[i],"up",
							"TopTable.tab", sep=""), sep="\t")
				write.table(topTable[[j]][[i]][which(topTable[[j]][[i]]$logFC<0),],
						file=paste(fname,names(topTable[[j]])[i],"dn",
							"TopTable.tab", sep=""), sep="\t")})}
			if(split=="f"){
				sapply(1:length(topTable[[j]]),function(i){
					write.table(topTable[[j]][[i]],

	file=paste(fname,names(topTable)[j],names(topTable[[j]])[i],
							"TopTable.tab", sep=""), sep="\t")})}})}
```

I.2.2.10 getMasterList

```
################
#getMasterList
################
#creates a list of probes shared by both array platforms used
#with the name of each type lined up next to each other
#so that the two platforms can be compiled into a single one.

getMasterList<-function(codeDir){
#read in the different lists
#UCSFGPR = UCSF array gpr file with index and probe names, ID, and sequence
#StanGPRwithUCSFGPR = Stanford array gpr file with index and probe names, ID,
#sequence, and corresponding UCSF probe names
	setwd(codeDir)
	UCSFGPR<-read.csv("y01_gpr_Seq.csv", colClasses="character")
	StanGPRwithUCSFGPR<-read.csv("StanGPRwithUCSFGPR.csv", colClasses="character")

#take all of the UCSFIDs that are shared by both, this will be me minimum
#set of probes that I can work with
	UCSFinter<-intersect(StanGPRwithUCSFGPR$UCSFID, UCSFGPR$ID)
	#8515 entries

#Use this unique list to determine the number of occurences of each ID in each list
	#alphabatize the GPR lists
		StanGPRwithUCSFGPR<-StanGPRwithUCSFGPR[order(StanGPRwithUCSFGPR$UCSFID),]
```

```
                #20000 probes
        UCSFGPR<-UCSFGPR[order(UCSFGPR$ID),]
                #21160 probes
        UCSFinter<-UCSFinter[order(UCSFinter)]
                #8515 probes

    #make column to keep track of the occurence # of that ID; fill with 0 for
    #those not in the intersecting list
        StanGPRwithUCSFGPR$occurence[1:length(StanGPRwithUCSFGPR$UCSFID)]<-0
        UCSFGPR$occurence[1:length(UCSFGPR$ID)]<-0

    #fill in the occurence column
        for (i in 1:length(UCSFinter)){
            #get all occurences of each unique sequence and put
            #indecies into a temp vector (y)
            y<-which(StanGPRwithUCSFGPR$UCSFID == UCSFinter[i])
            #each occurence of a name (index given by y) is given its
            #corresponding occurence number
            #(first index in the list gets a 1, second gets a 2, etc.)
            StanGPRwithUCSFGPR$occurence[y] <- 1:length(y)}

        for (i in 1:length(UCSFinter)){
            #get all occurence of each unique sequence and put
            #indecies into a temp vector (y)
            y<-which(UCSFGPR$ID == UCSFinter[i])
            #each occurence of a name (index given by y) is given its
            #corresponding occurence number
            #(first index in the list gets a 1, second gets a 2, etc.)
            UCSFGPR$occurence[y] <- 1:length(y)}

#remove all the spots that just occcur once
    #get the names for all of the spots that occur once
    loners<-setdiff(StanGPRwithUCSFGPR$UCSFID[StanGPRwithUCSFGPR$occurence == 1],
            StanGPRwithUCSFGPR$UCSFID[StanGPRwithUCSFGPR$occurence == 2])

    #intialize the new object
        StanGPRwithUCSFGPRNoLoners<-StanGPRwithUCSFGPR
    #remove all the loners one at a time
        for(i in 1:length(loners)){
            print(which(StanGPRwithUCSFGPRNoLoners$UCSFID == loners[i]))
            #take rows not matching the current loner
            StanGPRwithUCSFGPRNoLoners<-StanGPRwithUCSFGPRNoLoners[which(
                                StanGPRwithUCSFGPRNoLoners$UCSFID != loners[i]),]
            print(which(StanGPRwithUCSFGPRNoLoners$UCSFID == loners[i]))
            print(loners[i])
            print(dim(StanGPRwithUCSFGPRNoLoners))}

    #intialize the new object
        UCSFGPRNoLoners<-UCSFGPR
    #remove all the loners one at a time
        for(i in 1:length(loners)){
            print(which(UCSFGPRNoLoners$ID == loners[i]))
            #take rows not matching the current loner
            UCSFGPRNoLoners<-UCSFGPRNoLoners[which(
                                    UCSFGPRNoLoners$ID != loners[i]),]
            print(which(UCSFGPRNoLoners$ID == loners[i]))
            print(loners[i])
            print(dim(UCSFGPRNoLoners))}

#take only the first two occurences of each probe
```

```
        StanGPRwithUCSFGPRtwoSpot<-StanGPRwithUCSFGPRNoLoners[
                StanGPRwithUCSFGPRNoLoners$occurence <= 2 &
                StanGPRwithUCSFGPRNoLoners$occurence != 0, ]
                #17024 probes

        UCSFGPRtwoSpot<-UCSFGPRNoLoners[
                UCSFGPRNoLoners$occurence <= 2 &
                UCSFGPRNoLoners$occurence != 0, ]
                #17024 probes

#make the master list; contains probes that are shared by both chips and the
#appropriate index from each chip is listed in it's own column
        masterList<-cbind(StanGPRwithUCSFGPRtwoSpot$StanName,
                                                StanGPRwithUCSFGPRtwoSpot$Index,
                                                UCSFGPRtwoSpot$ID,
                                                UCSFGPRtwoSpot$Index)
        colnames(masterList)<-c("StanName", "StanIndex",
                        "UCSFID", "UCSFIndex")

        return(masterList)}
```

I.2.2.11 probeHomogenizeRG

```
################
#probeHomogenizeRG
################
#probeHomogenize - to allow utilization of arrays printed on two
#different platforms; to integrate RG objects

probeHomogenizeRG<-function(RGa, RGb, masterList){

#figure out number ID of columns that have the desired information
        colsa<-c(which(names(RGa)=="weights"),
                        which(names(RGa)=="R"),
                        which(names(RGa)=="G"),
                        which(names(RGa)=="genes"))

        colsb<-c(which(names(RGb)=="weights"),
                        which(names(RGb)=="R"),
                        which(names(RGb)=="G"),
                        which(names(RGb)=="genes"))

#create an RG object of the correct size
        RGam<-RGa[1:dim(masterList)[1],]
        RGbm<-RGb[1:dim(masterList)[1],]

#fill it in with probes specified by indecies in the masterList,
#which point back to row numbers in the gpr files
        for(i in 1:dim(masterList)[1]){
                #pull the RG row specified by the masterList index,
                #in the RG column/subobject specified by cols
                for(j in colsa){
                        RGam[[j]][i,]<-RGa[[j]][as.numeric(masterList[i,2]),]}}

        for(i in 1:dim(masterList)[1]){
                for(j in colsb){
                        RGbm[[j]][i,]<-RGb[[j]][as.numeric(masterList[i,4]),]}}

#give the lists the same names (replace StanNames with UCSFIDs)
#and stitch them up
        RGam$genes$Name<-RGbm$genes$Name
```

```
        RGam$genes$ID<-RGbm$genes$ID

        RG<-cbind(RGam, RGbm)

        return(RG)}
```

I.2.2.12 probeHomogenizeMA

```
################
#probeHomogenizeMA
################
#allows utilization of arrays printed on two different platforms;
#integrates MA objects

probeHomogenizeMA<-function(MAa, MAb, masterList){

#figure out number ID of columns that have the desired information
        colsa<-c(which(names(MAa)=="weights"),
                        which(names(MAa)=="M"),
                        which(names(MAa)=="A"),
                        which(names(MAa)=="genes"))

        colsb<-c(which(names(MAb)=="weights"),
                        which(names(MAb)=="M"),
                        which(names(MAb)=="A"),
                        which(names(MAb)=="genes"))

#create an object of the correct size
        MAam<-MAa[1:dim(masterList)[1],]
        MAbm<-MAb[1:dim(masterList)[1],]

#fill it in with probes specified by indecies in the masterList (run through these with i),
#which point back to row numbers in the gpr files
        for(i in 1:dim(masterList)[1]){
                for(j in colsa){
                        MAam[[j]][i,]<-MAa[[j]][as.numeric(masterList[i,2]),]}}
        for(i in 1:dim(masterList)[1]){
                for(j in colsb){
                        MAbm[[j]][i,]<-MAb[[j]][as.numeric(masterList[i,4]),]}}

#give the lists the same names (replace StanNames with UCSFIDs) and stitch them up
        MAam$genes$Name<-MAbm$genes$Name
        MAam$genes$ID<-MAbm$genes$ID

        MA<-cbind(MAam, MAbm)


#Add the block, row, and column info from the y01 arrays
#read through the MA gene names every other one and pull
#out the first two occurences

        for(i in seq(1,dim(MA)[1],by=2)){
                MA$genes$Block[i]<-MAb$genes$Block[which(MA$genes$ID[i]==MAb$genes$ID)[1]]
                MA$genes$Block[i+1]<-MAb$genes$Block[which(MA$genes$ID[i+1]==MAb$genes$ID)[2]]
                MA$genes$Row[i]<-MAb$genes$Row[which(MA$genes$ID[i]==MAb$genes$ID)[1]]
                MA$genes$Row[i+1]<-MAb$genes$Row[which(MA$genes$ID[i+1]==MAb$genes$ID)[2]]
                MA$genes$Column[i]<-MAb$genes$Column[which(MA$genes$ID[i]==MAb$genes$ID)[1]]
                MA$genes$Column[i+1]<-MAb$genes$Column[which(MA$genes$ID[i+1]==MAb$genes$ID)[2]]}

        return(MA)}
```

I.2.2.13 qualityPlots

```
################
#qualityPlots
################
#makes a series of plots to assess array quality

qualityPlots <- function(RG, doBkg, etype, arrayNames, groupList,
        intensityCutoff, chipType, dir){

#background plots
        if(doBkg){

#number of flags
        print("numFlags...")
        getData<-function(f){
                numCols<-dim(f$R)[2]
                flagged<-mat.or.vec(numCols,1)

                for(i in 1:numCols){
                        flagged[i]<-length(which(f$weights[,i]<1))}

                flagged<-data.frame(FlaggedSpots=flagged,
                        row.names=as.character(f$targets[,1]))
                return(flagged)}

        #get the data
        flags<-getData(RG)

        par(mfrow=c(1,2), family = "mono")
                cols<-c(sample(c("gray75"), size=3, replace=TRUE),
                        sample(c("gray25"), size=3, replace=TRUE))

        #plot the unfiltered data
        bp1<-barplot(flags[,1],names.arg=rownames(flags), col=cols,
                        ylim=c(0,max(flags[,1])+0.1*max(flags[,1])),
                        main="Manually Flagged Spots", ylab="Number of Flagged Spots",
                        cex.names=0.5, las=3)
        text(bp1, flags[,1],labels= flags[,1], pos=3, cex=0.5)

        #filter the data
        setwd(dir)
        if(chipType=="y01"){
                UCSFGPRtype<-read.csv("UCSFGPRtype.csv", colClasses="character")
                RG$genes$Type<-UCSFGPRtype[,2]
                RGf<-RG
                RGf$weights<-modifyWeights(RG$weights, RG$genes$Type,
                        c("CAS","CDP","CMM","CMO","EMPTY","TAG","0"),c(0.01))}
        if(chipType=="yOligo"){
                stanGlist<-read.csv("Stanford_Godlist_6_19_07_2.csv", colClasses="character")
                stanGlist$Index<-as.numeric(stanGlist$Index)

                stanGlistIndex<-read.csv("yOligo_StanGlist_Index.csv", colClasses="character")
                stanGlistIndex$CorStanGlistIndex<-as.numeric(stanGlistIndex$CorStanGlistIndex)
                #throws a warning about NA introduction;
                #this is for the small number (5) of features
                #that are on the gpr file and not on the godlist,
                #so they don't have a corresponding index

                RG$genes$StanGlistIndex<-stanGlistIndex$CorStanGlistIndex
```

```
                RG$genes$Type<-lapply(
                RG$genes$StanGlistIndex,
                function(i){stanGlist$TYPE[which(stanGlist$Index==i)]})
                        RG$genes$Type<-as.character(RG$genes$Type)

                RGf<-RG
                RGf$weights<-modifyWeights(RG$weights, RG$genes$Type,
                        c("CONTROL","MARKER","EMPTY"),c(0.01))}

        #get the flagged data
        flagsf<-getData(RGf)

        #plot the filtered data
        setwd(dir)
        bp2<-barplot(flagsf[,1],names.arg=rownames(flagsf), col=cols,
                        ylim=c(0,max(flagsf[,1])+0.1*max(flagsf[,1])),
                        main="Total Flagged Spots", ylab="Number of Flagged Spots",
                        cex.names=0.5, las=3)
        text(bp2, flagsf[,1],labels= flagsf[,1], pos=3, cex=0.5)

#get ranges for use in background plots

        bkg <- c(RG$Gb,RG$Rb)
        log.bkg <- log2(bkg)
        boxData.bkg<-boxplot(log.bkg, plot=FALSE)
        low.bkg<-boxData.bkg$stats[1,1] #extreme of lower whisker
        high.bkg<-boxData.bkg$stats[5,1] #extreme of upper whisker

#background imagePlots
        print("imagePlots...")
        #look for damage to the array
        #overall background intensity should be similar between the arrays
        #saturation limits are set by boxstats
        par(mfrow=c(2,2))
        sapply(1:dim(RG)[2], function(i) {
                imageplot(log2(RG$Gb[,i]), RG$printer,
                        main = paste(arrayNames[i], "Gb", sep=" "),
                zlim = c(low.bkg,high.bkg))
                imageplot(log2(RG$Rb[,i]), RG$printer, main =
                        paste(arrayNames[i], "Rb", sep=" "),
                zlim = c(low.bkg,high.bkg))})

#background boxplots
        print("background boxplots...")
        #want the background to be more or less homogeneous
        #both in it's distribution (the size of the box)
        #and it's median (where the box is centered)
        #get rid of the zero values
        RG$Gb[which(RG$Gb==0)]<-median(RG$Gb)
        RG$Rb[which(RG$Rb==0)]<-median(RG$Rb)

        cols<-mat.or.vec(1,dim(RG)[2])
        cols[groupList[["WT"]]]<-"tan"
        cols[groupList[["e4G1"]]]<-"royalblue"
        cols[groupList[["e4G2"]]]<-"orange"

        par(mfrow=c(1,2))
        boxplot(as.data.frame(log2(RG$Gb)), main = "Gb",
                names=arrayNames, las=3, cex.axis=0.5, col=cols,
                ylim = c(low.bkg-0.25*low.bkg, high.bkg+0.25*high.bkg))
```

```
        boxplot(as.data.frame(log2(RG$Rb)), main = "Rb",
                names=arrayNames, las=3, cex.axis=0.5, col=cols,
                ylim = c(low.bkg-0.25*low.bkg, high.bkg+0.25*high.bkg))}#doBkg

#foreground ranges
        RG$G[which(RG$G==0)]<-median(RG$G)
        RG$R[which(RG$R==0)]<-median(RG$R)

        RGlims <- c(min(log2(c(RG$R,RG$G)),na.rm=TRUE),
                                max(log2(c(RG$R,RG$G)),na.rm=TRUE))

        MA<-MA.RG(RG, bc.method="none")

        Mlims <- c(min(MA$M,na.rm=TRUE)-0.05*min(MA$M,na.rm=TRUE),
        max(MA$M,na.rm=TRUE)+0.01*max(MA$M,na.rm=TRUE))
        Alims <- c(min(MA$A,na.rm=TRUE)-0.05*min(MA$A,na.rm=TRUE),
                               max(MA$A,na.rm=TRUE)+0.01*max(MA$A,na.rm=TRUE))

#foreground densityPlots
        print("foreground density...")
        #these are matricies; density data in rows, arrays in cols
        G.densities <- sapply(1:dim(RG)[2], function(i){density(
                log2(RG$G[,i]),n=dim(RG)[1],na.rm=TRUE)})
        R.densities <- sapply(1:dim(RG)[2], function(i){density(
                log2(RG$R[,i]),n=dim(RG)[1],na.rm=TRUE)})

        yGlimits <- c(min(unlist(G.densities[2,]))-
                                        0.05*min(unlist(G.densities[2,])),
                                max(unlist(G.densities[2,]))+0.05*
                                        max(unlist(G.densities[2,])))
        yRlimits <- c(min(unlist(R.densities[2,]))-
                                        0.05*min(unlist(R.densities[2,])),
                                max(unlist(R.densities[2,]))+0.05*
                                        max(unlist(R.densities[2,])))

#type1 = PvP/TvT arrays (which have a reference involved)
        if(etype=="PvP" | etype=="TvT"){

#color by  strain - to invenstigate strain effects
        #color scheme for array hybs
        #WT=tan, d4G1=blue, d4G2=orange, ref=azure
        #bio1=light shades, bio2=dark shades
        rcols<-mat.or.vec(1,dim(RG)[2])
        gcols<-mat.or.vec(1,dim(RG)[2])

        for(i in seq(1,length(groupList[["e4G1"]]),4)){
                                rcols[groupList[["WT"]][i]]<-"tan4"
                                rcols[groupList[["WT"]][i+1]]<-"azure3"
                                rcols[groupList[["WT"]][i+2]]<-"saddlebrown"
                                rcols[groupList[["WT"]][i+3]]<-"azure4"

                                rcols[groupList[["e4G1"]][i]]<-"royalblue"
                                rcols[groupList[["e4G1"]][i+1]]<-"azure3"
                                rcols[groupList[["e4G1"]][i+2]]<-"royalblue4"
                                rcols[groupList[["e4G1"]][i+3]]<-"azure4"

                                rcols[groupList[["e4G2"]][i]]<-"orange"
                                rcols[groupList[["e4G2"]][i+1]]<-"azure3"
                                rcols[groupList[["e4G2"]][i+2]]<-"red3"
                                rcols[groupList[["e4G2"]][i+3]]<-"azure4"}
```

```
        for(i in seq(1,length(groupList[["e4G1"]]),4)){
                                gcols[groupList[["WT"]][i]]<-"azure3"
                                gcols[groupList[["WT"]][i+1]]<-"tan4"
                                gcols[groupList[["WT"]][i+2]]<-"azure4"
                                gcols[groupList[["WT"]][i+3]]<-"saddlebrown"

                                gcols[groupList[["e4G1"]][i]]<-"azure3"
                                gcols[groupList[["e4G1"]][i+1]]<-"royalblue"
                                gcols[groupList[["e4G1"]][i+2]]<-"azure4"
                                gcols[groupList[["e4G1"]][i+3]]<-"royalblue4"

                                gcols[groupList[["e4G2"]][i]]<-"azure3"
                                gcols[groupList[["e4G2"]][i+1]]<-"orange"
                                gcols[groupList[["e4G2"]][i+2]]<-"azure4"
                                gcols[groupList[["e4G2"]][i+3]]<-"red3"}

        #separate plots for different strains
                par(mfrow=c(2,2))
                for(j in 1:length(groupList)){
                plot(x=NULL, type="l",main=paste(names(groupList)[j], "Fgr", sep = " "),
                         xlab="Intensity",ylab="Density",cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(groupList[[j]], function(i){lines(
                        density(log2(RG$R[,i]),na.rm=TRUE),col=rcols[i])})
                sapply(groupList[[j]], function(i){lines(
                        density(log2(RG$G[,i]),na.rm=TRUE),col=gcols[i])})
                legend(x="topright",lty=1,cex=0.65,text.col="black",
                           legend=c(paste(names(groupList)[j]," Bio1"),paste(
                                       names(groupList)[j]," Bio2"),"Ref Bio1", "Ref Bio2"),
                           col=c(rcols[j],rcols[j+6],rcols[j+3],rcols[j+9]))}

        #all strains on the same plot
                plot(x=NULL, type="l",main=paste("Fgr", sep = " "),xlab="Intensity",
                        ylab="Density", cex.main=1.5,font.main=2, xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=rcols[i])})
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$G[,i]),na.rm=TRUE),col=gcols[i])})

#color by  dye - to investigate dye effects
        #color scheme for array hybs
        #dark=Bio1, light=Bio2
                redcols <-c(rep(c(rep("red", times=6),rep("darkred", times=6)),
                        (dim(RG)[2]/12)))
                grncols <-c(rep(c(rep("chartreuse2", times=6),rep("chartreuse4", times=6)),
                        (dim(RG)[2]/12)))

        #different strains on different plots
                par(mfrow=c(2,2))
                for(j in 1:length(groupList)){
                plot(x=NULL, type="l",main=paste(names(groupList)[j], "Bkg", sep = " "),
                         xlab="Intensity",ylab="Density",cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(groupList[[j]], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=redcols[i])})
                sapply(groupList[[j]], function(i){lines(density(
```

```
                        log2(RG$G[,i]),na.rm=TRUE),col=grncols[i])})
                mtext("Light=Bio1, Dark=Bio2", side=3, line=-1.5, col="black",cex=0.75)}

        #all strains on the same plot
                plot(x=NULL, type="l",main=paste("Bkg", sep = " "),xlab="Intensity",
                        ylab="Density",
                         cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=redcols[i])})
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$G[,i]),na.rm=TRUE),col=grncols[i])})
                mtext("Light=Bio1, Dark=Bio2", side=3, line=-1.5, col="black",cex=0.75)

#use plotDensities(), to double check
                par(mfrow=c(2,2))
                for(j in 1:length(groupList)){
                plotDensities(RG, arrays=groupList[[j]])
                                mtext(names(groupList)[j], side=3, cex=1.5, font=2,col="black")
                                mtext("Dark=Bio1, Light=Bio2", side=3, line=-
1.5,col="black",cex=0.75)}
                plotDensities(RG)
                mtext("All Data", side=3, cex=1.5, font=2,col="black")

#each array gets it's own plot - to track down the bad apples
                par(mfrow=c(3,4))
                for(i in 1:dim(RG)[2]){
                plot(density(log2(RG$R[,i]),na.rm=TRUE), col="red", type="l",
                        main=arrayNames[i],
                         xlab="Intensity",ylab="Density",cex.main=1,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))}
                par(mfrow=c(3,4))
                for(i in 1:dim(RG)[2]){
                plot(density(log2(RG$G[,i]),na.rm=TRUE), col="chartreuse2", type="l",
                        main=arrayNames[i],
                         xlab="Intensity",ylab="Density",cex.main=1,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))}}#type1

if(etype=="PvT"){
#color by  strain - to invenstigate strain effects
        #color scheme for array hybs
        #WT=tan, d4G1=blue, d4G2=orange,
        #poly=light shades, total=dark shades; no distinction made between bio dupes here
                rcols<-mat.or.vec(1,dim(RG)[2])
                gcols<-mat.or.vec(1,dim(RG)[2])

                rcols[groupList[["WT"]]]<-c("tan","tan4")
                rcols[groupList[["e4G1"]]]<-c("royalblue","royalblue4")
                rcols[groupList[["e4G2"]]]<-c("orange","red3")

                gcols[groupList[["WT"]]]<-c("tan4","tan")
                gcols[groupList[["e4G1"]]]<-c("royalblue4","royalblue")
                gcols[groupList[["e4G2"]]]<-c("red3","orange")

        #separate plots for the different strains
                par(mfrow=c(2,2))
                for(j in 1:length(groupList)){
```

```
                plot(x=NULL, type="l",main=paste(names(groupList)[j], "Bkg", sep = " "),
                         xlab="Intensity",ylab="Density",cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(groupList[[j]], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=rcols[i])})
                sapply(groupList[[j]], function(i){lines(density(
                        log2(RG$G[,i]),na.rm=TRUE),col=gcols[i])})
                legend(x="topright",lty=1,cex=0.5,text.col="black",
                           legend=c(paste(names(groupList)[j]," Poly"),
                           paste(names(groupList)[j]," Total")),
                           col=c(rcols[j],rcols[j+3]))}

        #all strains on the same plot
                plot(x=NULL, type="l",main=paste("Bkg", sep = " "),xlab="Intensity",
                        ylab="Density",
                         cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=rcols[i])})
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$G[,i]),na.rm=TRUE),col=gcols[i])})

#color by  dye - to invenstigate dye effects
        #color scheme for array hybs
        #poly=light shades, total=dark shades; no distinction made between bio dupes here
                redcols <-c(rep(c(rep("red", times=3),rep("darkred", times=3)),
                        (dim(RG)[2]/6)))
                grncols <-c(rep(c(rep("chartreuse2", times=3),rep("chartreuse4", times=3)),
                        (dim(RG)[2]/6)))

                #plots for different strains
                par(mfrow=c(2,2))
                for(j in 1:length(groupList)){
                plot(x=NULL, type="l",main=paste(names(groupList)[j], "Bkg", sep = " "),
                         xlab="Intensity",ylab="Density",cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(groupList[[j]], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=redcols[i])})
                sapply(groupList[[j]], function(i){lines(density(
                        log2(RG$G[,i]),na.rm=TRUE),col=grncols[i])})
                mtext("Light=Poly, Dark=Total", side=3, line=-1.5, col="black",cex=0.75)}

                #all strains on one plot
                plot(x=NULL, type="l",main=paste("Bkg", sep = " "),xlab="Intensity",
                        ylab="Density",
                         cex.main=1.5,font.main=2,
                         xlim = RGlims,
                         ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$R[,i]),na.rm=TRUE),col=redcols[i])})
                sapply(1:dim(RG)[2], function(i){lines(density(
                        log2(RG$G[,i]),na.rm=TRUE),col=grncols[i])})
                mtext("Light=Poly, Dark=Total", side=3, line=-1.5, col="black",cex=0.75)

#use plotDensities(), to double check
                par(mfrow=c(2,2))
                for(j in 1:length(groupList)){
```

```
            plotDensities(RG, arrays=groupList[[j]])
                                    mtext(names(groupList)[j], side=3, cex=1.5,
font=2,col="black")}
            plotDensities(RG)
                                    mtext("All Data", side=3, cex=1.5, font=2,col="black")

#each array gets it's own plot - to track down the bad apples
            par(mfrow=c(3,4))
            for(i in 1:dim(RG)[2]){
            plot(density(log2(RG$R[,i]),na.rm=TRUE), col="red", type="l",
                   main=arrayNames[i],
                    xlab="Intensity",ylab="Density",cex.main=1,font.main=2,
                    xlim = RGlims,
                    ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))}
            for(i in 1:dim(RG)[2]){
            plot(density(log2(RG$G[,i]),na.rm=TRUE), col="chartreuse2",
                   type="l",main=arrayNames[i],
                    xlab="Intensity",ylab="Density",cex.main=1,font.main=2,
                    xlim = RGlims,
                    ylim = c(min(yGlimits, yRlimits), max(yGlimits, yRlimits)))}}

#box plots
    print("foreground box...")
    #are the boxes all over the place?
            cols<-mat.or.vec(1,dim(RG)[2])
            cols[groupList[["WT"]]]<-"tan"
            cols[groupList[["e4G1"]]]<-"royalblue"
            cols[groupList[["e4G2"]]]<-"orange"

        par(mfrow=c(1,2))
            boxplot(as.data.frame(log2(RG$G)), main = paste("log2(Green)", sep = " "),
                names=arrayNames, col=cols,
                las=3, cex.axis=0.5, ylim = RGlims)
        boxplot(as.data.frame(log2(RG$R)), main = paste("log2(Red)", sep = " "),
                       names=arrayNames, col=cols,
                       las=3, cex.axis=0.5, ylim = RGlims)

            par(mfrow=c(1,1))
        boxplot(MA$M~col(MA$M), main = paste("M-value", sep = " "),
                       names=arrayNames, col=cols,
                       las=3, cex.axis=0.5, ylim = Mlims)


#MA plots
        print("MAplots...")
        #look for a good spread of the data throughout the intensity range.
        par(mfcol=c(3,2),col=rgb(56,56,56,20,maxColorValue=255),pch=16)
        sapply(1:dim(RG)[2], function(i) {
                plotMA(MA, array=i, ylim = Mlims, xlim = Alims,
                          main=paste(arrayNames[i],sep=" "),
                          status=MA$genes$Type,
                          values=c("CAS","CDP","CMM","CMO","EMPTY","TAG","OCH","0"),
                          col=rgb(0,0,0,0,maxColorValue=255),
                          legend=FALSE,
                          xlab="A (Ave log2(intensity))",
                          ylab="M (log2(R/G))")
                          abline(h = 0, col=rgb(20,20,20,100,maxColorValue=255), lwd=2)
        })

        ###################################
```

```
        #technical replicate correlation plots
        ###################################
        print("techCor plots...")
        par(mfrow=c(2,2), col=rgb(56,56,56,20,maxColorValue=255),
                pch=20,cex=0.25,cex.lab=1.5,cex.main=3)
        sapply(1:length(groupList),function(j){
                sapply(seq(1,length(groupList[[j]]),2), function(i){
                        plot(MA$M[,groupList[[j]][i]][which(
                                        MA$weights[,groupList[[j]][i]]==1 &
                                        MA$weights[,groupList[[j]][(i+1)]]==1)],
                                -MA$M[,groupList[[j]][(i+1)]][which(
                                        MA$weights[,groupList[[j]][i]]==1 &
                                        MA$weights[,groupList[[j]][(i+1)]]==1)],
                                ylim = Mlims, xlim = Mlims, asp=1,
                                main=paste(names(groupList)[j],"Tech Dupe Correlation",sep=" "),
                                xlab=paste(arrayNames[groupList[[j]][i]],"M (log2(R/G))"),
                                ylab=paste("-",arrayNames[groupList[[j]][(i+1)]],"M (log2(R/G))"))
                                abline(0,1,col=rgb(20,20,20,100,maxColorValue=255), lwd=2)

                        mtext(paste("cor:",format(
                                cor(MA$M[,groupList[[j]][i]][which(
                                                MA$weights[,groupList[[j]][i]]==1 &
                                                MA$weights[,groupList[[j]][(i+1)]]==1)],
                                        -MA$M[,groupList[[j]][(i+1)]][which(
                                                MA$weights[,groupList[[j]][i]]==1 &
                                                MA$weights[,groupList[[j]][(i+1)]]==1)]),
                                digits=3),sep=" "),side=1, line=-1.5, adj=1 ,col="black",cex=1,
font=2)

                        mtext(paste(format(
                                length(MA$M[,groupList[[j]][i]][which(
                                                MA$weights[,groupList[[j]][i]]==1 &
                                                MA$weights[,groupList[[j]][(i+1)]]==1)]),
                                digits=3),"points",sep=" "),side=1, line=-1.5, adj=0 ,
                                col="black",cex=1, font=2)})})

#technical replicate correlation plots       with intensity cutoff
        if(intensityCutoff){
        par(mfrow=c(2,2), col=rgb(56,56,56,20,maxColorValue=255),pch=20,cex=0.25,
                        cex.lab=1.5,cex.main=3)
        sapply(1:length(groupList),function(j){
                sapply(seq(1,length(groupList[[j]]),2), function(i){
                        plot(MA$M[,groupList[[j]][i]][which(
                                        MA$weights[,groupList[[j]][i]]==1 &
                                        MA$weights[,groupList[[j]][(i+1)]]==1 &
                                        MA$A[,groupList[[j]][i]]>intensityCutoff &
                                        MA$A[,groupList[[j]][(i+1)]]>intensityCutoff)],
                                -
MA$M[,groupList[[j]][(i+1)]][which(MA$weights[,groupList[[j]][i]]==1 &
                                        MA$weights[,groupList[[j]][(i+1)]]==1 &
                                        MA$A[,groupList[[j]][i]]>intensityCutoff &
                                        MA$A[,groupList[[j]][(i+1)]]>intensityCutoff)],
                                ylim = Mlims, xlim = Mlims, asp=1,
                                main=paste(names(groupList)[j],"Tech Dupe Cor Int Cutoff",sep=" "),
                                xlab=paste(arrayNames[groupList[[j]][i]],"M (log2(R/G))"),
                                ylab=paste("-",arrayNames[groupList[[j]][(i+1)]],"M (log2(R/G))"))
                        abline(0,1,col=rgb(20,20,20,100,maxColorValue=255), lwd=2)
                        mtext(paste("cor:",format(
                                cor(MA$M[,groupList[[j]][i]][which(
                                                MA$weights[,groupList[[j]][i]]==1 &
```

```
                                        MA$weights[,groupList[[j]][(i+1)]]==1 &
                                        MA$A[,groupList[[j]][i]]>intensityCutoff &
                                        MA$A[,groupList[[j]][(i+1)]]>intensityCutoff)],
                                -
MA$M[,groupList[[j]][(i+1)]][which(MA$weights[,groupList[[j]][i]]==1 &
                                        MA$weights[,groupList[[j]][(i+1)]]==1 &
                                        MA$A[,groupList[[j]][i]]>intensityCutoff &
                                        MA$A[,groupList[[j]][(i+1)]]>intensityCutoff)]),
                                digits=3),sep=" "),side=1, line=-1.5, adj=1
,col="black",cex=1, font=2)
                        mtext(paste(format(
                                length(MA$M[,groupList[[j]][i]][which(
                                        MA$weights[,groupList[[j]][i]]==1 &
                                        MA$weights[,groupList[[j]][(i+1)]]==1 &
                                        MA$A[,groupList[[j]][i]]>intensityCutoff &
                                        MA$A[,groupList[[j]][(i+1)]]>intensityCutoff)]),
                                digits=3),"points",sep=" "),side=1, line=-1.5, adj=0 ,
                                col="black",cex=1, font=2)})})}}
```

I.2.2.14 y01eIF4GspotData

```
################
#y01eIF4GspotData
################
#to look at 4G spot data as a control to see if it correlates with strain
#as a sanity check
        #input: RG object
        #output: Barchart

y01eIF4GspotData<-function(RGlist,etype,targets,groupList,
        arrayNames,chipType){
        if(chipType=="y01"){   e4G1ID="YGR162W"
                                                e4G2ID="YGL049C"}

        if(chipType=="yOligo"){       e4G1ID="YGR162W_ORF"
                                                e4G2ID="YGL049C_ORF"}


        getData<-function(x,combo){
        #get the raw spot data
        #average within array spots
        R1<-x$R[which(x$genes$ID==e4G1ID),]
        R1<-sapply(1:dim(R1)[2],function(i){mean(R1[,i])})
        G1<-x$G[which(x$genes$ID==e4G1ID),]
        G1<-sapply(1:dim(G1)[2],function(i){mean(G1[,i])})
        M1<-log(R1/G1, base=2)

        R2<-x$R[which(x$genes$ID==e4G2ID),]
        R2<-sapply(1:dim(R2)[2],function(i){mean(R2[,i])})
        G2<-x$G[which(x$genes$ID==e4G2ID),]
        G2<-sapply(1:dim(G2)[2],function(i){mean(G2[,i])})
        M2<-log(R2/G2, base=2)

        if(combo==TRUE){
        #combine tech reps
        #eIF4G1 data
        #experimental/poly channel
        #numeric vector of WT, G1, and G2 data (all bio and tech reps combined)
        eIF4G1Edata<-
                c(WT=mean(c(R1[groupList[["WT"]][
                                        seq(1,length(groupList[["WT"]]),2)]],
```

```
                        G1[groupList[["WT"]][
                                    seq(2,length(groupList[["WT"]]),2)]]),na.rm=TRUE),
          G1=mean(c(R1[groupList[["e4G1"]][
                                    seq(1,length(groupList[["e4G1"]]),2)]],
                        G1[groupList[["e4G1"]][
                                    seq(2,length(groupList[["e4G1"]]),2)]]),na.rm=TRUE),
          G2=mean(c(R1[groupList[["e4G2"]][
                                    seq(1,length(groupList[["e4G2"]]),2)]],
                        G1[groupList[["e4G2"]][
                                    seq(2,length(groupList[["e4G2"]]),2)]]),na.rm=TRUE))
#reference/total channel
eIF4G1Rdata<-
        c(WT=mean(c(G1[groupList[["WT"]][
                                    seq(1,length(groupList[["WT"]]),2)]],
                        R1[groupList[["WT"]][
                                    seq(2,length(groupList[["WT"]]),2)]]),na.rm=TRUE),
          G1=mean(c(G1[groupList[["e4G1"]][
                                    seq(1,length(groupList[["e4G1"]]),2)]],
                        R1[groupList[["e4G1"]][
                                    seq(2,length(groupList[["e4G1"]]),2)]]),na.rm=TRUE),
          G2=mean(c(G1[groupList[["e4G2"]][
                                    seq(1,length(groupList[["e4G2"]]),2)]],
                        R1[groupList[["e4G2"]][
                                    seq(2,length(groupList[["e4G2"]]),2)]]),na.rm=TRUE))
#M-values
eIF4G1Mdata<-
        c(WT=mean(c(M1[groupList[["WT"]][
                                    seq(1,length(groupList[["WT"]]),2)]],
                       -M1[groupList[["WT"]][
                                    seq(2,length(groupList[["WT"]]),2)]]),na.rm=TRUE),
          G1=mean(c(M1[groupList[["e4G1"]][
                                    seq(1,length(groupList[["e4G1"]]),2)]],
                         -M1[groupList[["e4G1"]][
                                    seq(2,length(groupList[["e4G1"]]),2)]]),na.rm=TRUE),
          G2=mean(c(M1[groupList[["e4G2"]][
                                    seq(1,length(groupList[["e4G2"]]),2)]],
                         -M1[groupList[["e4G2"]][
                                    seq(2,length(groupList[["e4G2"]]),2)]]),na.rm=TRUE))

eIF4G1data<-rbind(eIF4G1Edata, eIF4G1Rdata, eIF4G1Mdata)

#eIF4G2 data
#experimental/poly channel
eIF4G2Edata<-
        c(WT=mean(c(R2[groupList[["WT"]][
                                    seq(1,length(groupList[["WT"]]),2)]],
                        G2[groupList[["WT"]][
                                    seq(2,length(groupList[["WT"]]),2)]]),na.rm=TRUE),
          G1=mean(c(R2[groupList[["e4G1"]][
                                    seq(1,length(groupList[["e4G1"]]),2)]],
                        G2[groupList[["e4G1"]][
                                    seq(2,length(groupList[["e4G1"]]),2)]]),na.rm=TRUE),
          G2=mean(c(R2[groupList[["e4G2"]][
                                    seq(1,length(groupList[["e4G2"]]),2)]],
                        G2[groupList[["e4G2"]][
                                    seq(2,length(groupList[["e4G2"]]),2)]]),na.rm=TRUE))
#reference/total channel
eIF4G2Rdata<-
        c(WT=mean(c(G2[groupList[["WT"]][
                                    seq(1,length(groupList[["WT"]]),2)]],
```

```
                        R2[groupList[["WT"]][
                                    seq(2,length(groupList[["WT"]]),2)]]),na.rm=TRUE),
          G1=mean(c(G2[groupList[["e4G1"]][
                                    seq(1,length(groupList[["e4G1"]]),2)]],
                        R2[groupList[["e4G1"]][
                                    seq(2,length(groupList[["e4G1"]]),2)]]),na.rm=TRUE),
          G2=mean(c(G2[groupList[["e4G2"]][
                                    seq(1,length(groupList[["e4G2"]]),2)]],
                        R2[groupList[["e4G2"]][
                                    seq(2,length(groupList[["e4G2"]]),2)]]),na.rm=TRUE))
#M-values
eIF4G2Mdata<-
        c(WT=mean(c(M2[groupList[["WT"]][
                                    seq(1,length(groupList[["WT"]]),2)]],
                       -M2[groupList[["WT"]][
                                    seq(2,length(groupList[["WT"]]),2)]]),na.rm=TRUE),
          G1=mean(c(M2[groupList[["e4G1"]][
                                    seq(1,length(groupList[["e4G1"]]),2)]],
                        -M2[groupList[["e4G1"]][
                                    seq(2,length(groupList[["e4G1"]]),2)]]),na.rm=TRUE),
          G2=mean(c(M2[groupList[["e4G2"]][
                                    seq(1,length(groupList[["e4G2"]]),2)]],
                        -M2[groupList[["e4G2"]][
                                    seq(2,length(groupList[["e4G2"]]),2)]]),na.rm=TRUE))

eIF4G2data<-rbind(eIF4G2Edata, eIF4G2Rdata, eIF4G2Mdata)

eIF4Gdata<-cbind(eIF4G1data, eIF4G2data)
return(eIF4Gdata)
}#combo
else{
        return(rbind(R1,G1,R2,G2)) #arrays in columns; 4rows: R1, G1, R2, G2
}
}#get data

#look at all arrays first
print("getting all array 4G spot data...")
eIF4GdataRGrawlist<-lapply(RGlist,getData,combo=FALSE)

        #color scheme for array hybs
        #WT=tan, d4G1=royalblue, d4G2=orange, ref=azure
        #bio1=light shades, bio2=dark shades
        print("getting all array 4G plot colors...")

        rcols<-mat.or.vec(1,dim(eIF4GdataRGrawlist[[1]])[2])
        gcols<-mat.or.vec(1,dim(eIF4GdataRGrawlist[[1]])[2])

        if(etype=="PvP" | etype=="TvT"){
        for(i in seq(1,length(groupList[["e4G1"]]),4)){
                                rcols[groupList[["WT"]][i]]<-"tan4"
                                rcols[groupList[["WT"]][i+1]]<-"azure3"
                                rcols[groupList[["WT"]][i+2]]<-"saddlebrown"
                                rcols[groupList[["WT"]][i+3]]<-"azure4"

                                rcols[groupList[["e4G1"]][i]]<-"royalblue"
                                rcols[groupList[["e4G1"]][i+1]]<-"azure3"
                                rcols[groupList[["e4G1"]][i+2]]<-"royalblue4"
                                rcols[groupList[["e4G1"]][i+3]]<-"azure4"

                                rcols[groupList[["e4G2"]][i]]<-"orange"
```

```
                                rcols[groupList[["e4G2"]][i+1]]<-"azure3"
                                rcols[groupList[["e4G2"]][i+2]]<-"red3"
                                rcols[groupList[["e4G2"]][i+3]]<-"azure4"}

        for(i in seq(1,length(groupList[["e4G1"]]),4)){
                                gcols[groupList[["WT"]][i]]<-"azure3"
                                gcols[groupList[["WT"]][i+1]]<-"tan"
                                gcols[groupList[["WT"]][i+2]]<-"azure4"
                                gcols[groupList[["WT"]][i+3]]<-"tan4"

                                gcols[groupList[["e4G1"]][i]]<-"azure3"
                                gcols[groupList[["e4G1"]][i+1]]<-"royalblue"
                                gcols[groupList[["e4G1"]][i+2]]<-"azure4"
                                gcols[groupList[["e4G1"]][i+3]]<-"royalblue4"

                                gcols[groupList[["e4G2"]][i]]<-"azure3"
                                gcols[groupList[["e4G2"]][i+1]]<-"orange"
                                gcols[groupList[["e4G2"]][i+2]]<-"azure4"
                                gcols[groupList[["e4G2"]][i+3]]<-"red3"}}
        if(etype=="PvT"){
                rcols[groupList[["WT"]]]<-c("tan","tan4")
                rcols[groupList[["e4G1"]]]<-c("royalblue","royalblue4")
                rcols[groupList[["e4G2"]]]<-c("orange","red3")

                gcols[groupList[["WT"]]]<-c("tan4","tan")
                gcols[groupList[["e4G1"]]]<-c("royalblue4","royalblue")
                gcols[groupList[["e4G2"]]]<-c("red3","orange")}

par(mfrow=c(2,2), family = "mono")
print("all array 4G spot plot...")

#4g1-spots
for(i in 1:length(eIF4GdataRGrawlist)){
 bp1<-barplot(as.numeric(sapply(1:dim(eIF4GdataRGrawlist[[i]])[2], function(j){
                        eIF4GdataRGrawlist[[i]][1:2,j]})),
                        ylim=c(0,max(eIF4GdataRGrawlist[[i]][1:2,]) +
                        0.25*max(eIF4GdataRGrawlist[[i]][1:2,])),

names.arg=as.character(sapply(1:length(RGlist[[1]]$targets[,1]),function(i){
                        rep(RGlist[[1]]$targets[i,1],2)})),
col=as.character(sapply(1:length(rcols),function(i){c(rcols[i],gcols[i])})),
                        main=paste("4G1 Spot",names(eIF4GdataRGrawlist)[i],sep=":"),
                        xlab="Strain", ylab="Intensity",
                        cex.names=0.75, las=3)
        text(bp1,as.numeric(sapply(1:dim(eIF4GdataRGrawlist[[i]])[2], function(j){
                                eIF4GdataRGrawlist[[i]][1:2,j]})),
        labels= format(as.numeric(sapply(1:dim(eIF4GdataRGrawlist[[i]])[2], function(j){
                                eIF4GdataRGrawlist[[i]][1:2,j]})), digits=1),
                        pos=3, cex=0.5)
if(etype=="PvP" | etype=="TvT"){
        legend(x="topleft",lty=1,cex=0.4,text.col="black",ncol=2,
                        legend=c("WT-1","G1-1","G2-1","Ref-1","WT-2","G1-2","G2-2","Ref-
2"),

col=c(rcols[1],rcols[2],rcols[3],rcols[4],rcols[7],rcols[8],rcols[9],rcols[10]))}
if(etype=="PvT"){
        legend(x="topleft",lty=1,cex=0.4,text.col="black",,ncol=2,
                                legend=c("WT-P","WT-T","G1-P","G1-T","G2-P","G2-T"),

col=c(rcols[1],gcols[1],rcols[2],gcols[2],rcols[3],gcols[3]))}}
```

```
        #4g2-spots
        for(i in 1:length(eIF4GdataRGrawlist)){
         bp1<-barplot(as.numeric(sapply(1:dim(eIF4GdataRGrawlist[[i]])[2], function(j){
                                    eIF4GdataRGrawlist[[i]][3:4,j]})),
                                    ylim=c(0,max(eIF4GdataRGrawlist[[i]][3:4,]) +
                                    0.25*max(eIF4GdataRGrawlist[[i]][3:4,])),

        names.arg=as.character(sapply(1:length(RGlist[[1]]$targets[,1]),function(i){
                                    rep(RGlist[[1]]$targets[i,1],2)})),

        col=as.character(sapply(1:length(rcols),function(i){c(rcols[i],gcols[i])})),
                                    main=paste("4G2
Spot",names(eIF4GdataRGrawlist)[i],sep=":"),
                                    xlab="Strain", ylab="Intensity",
                                    cex.names=0.75, las=3)
                text(bp1,as.numeric(sapply(1:dim(eIF4GdataRGrawlist[[i]])[2], function(j){
                                    eIF4GdataRGrawlist[[i]][3:4,j]})),
                labels= format(as.numeric(sapply(1:dim(eIF4GdataRGrawlist[[i]])[2], function(j){
                                    eIF4GdataRGrawlist[[i]][3:4,j]})), digits=1),
                                pos=3, cex=0.5)
        if(etype=="PvP" | etype=="TvT"){
                legend(x="topleft",lty=1,cex=0.4,text.col="black",ncol=2,
                                    legend=c("WT-1","G1-1","G2-1","Ref-1","WT-2","G1-2","G2-
2","Ref-2"),

        col=c(rcols[1],rcols[2],rcols[3],rcols[4],rcols[7],rcols[8],rcols[9],rcols[10]))}
        if(etype=="PvT"){
                legend(x="topleft",lty=1,cex=0.4,text.col="black",,ncol=2,
                                    legend=c("WT-P","WT-T","G1-P","G1-T","G2-P","G2-T"),

        col=c(rcols[1],gcols[1],rcols[2],gcols[2],rcols[3],gcols[3]))}}

        #combine duplicates
        print("getting combo 4G spot data...")

        eIF4GdataRGlist<-lapply(RGlist,getData,combo=TRUE)

        cols1<-c(rgb(27,66,123,maxColorValue=255),rgb(128,128,0,maxColorValue=255))
        cols2<-c(rgb(150,1,12,maxColorValue=255), rgb(128,128,0,maxColorValue=255))

        if(etype=="PvP" | etype=="TvT"){
        barnames<-c(rep(c("WT","WT-Ref","e4G1","G1-Ref","e4G2","G2-Ref"),2))}
        if(etype=="PvT"){
        barnames<-c(rep(c("WTp","WTt","e4G1p","e4G1t","e4G2p","e4G1t"),2))}

        print("combo 4G spot plots...")

        par(mfrow=c(2,2), family = "mono")
        for(i in 1:length(eIF4GdataRGrawlist)){
                        barplot(as.numeric(sapply(1:dim(eIF4GdataRGlist[[i]])[2], function(j){
                                            eIF4GdataRGlist[[i]][1:2,j]})),
                                            ylim=c(0,max(eIF4GdataRGlist[[i]][1:2,]) +
                                        0.25*max(eIF4GdataRGlist[[i]][1:2,])),
                                            names.arg=barnames,col=c(cols1, cols1, cols1, cols2,
cols2, cols2),
                                            main=names(eIF4GdataRGrawlist)[i], xlab="Strain",
ylab="Intensity",
                                            cex.names=0.75, las=3)
                        text(bp1,as.numeric(sapply(1:dim(eIF4GdataRGlist[[i]])[2], function(j){
```

```
                                        eIF4GdataRGlist[[i]][1:2,j]})),
                labels= format(as.numeric(sapply(1:dim(eIF4GdataRGlist[[i]])[2],
function(j){
                                        eIF4GdataRGlist[[i]][1:2,j]})), digits=1),
                                        pos=3, cex=0.6)
        if(etype=="PvP" | etype=="TvT"){
                legend(x="topright",lty=1,cex=0.6,text.col="black", ncol=2,
                                        legend=c("G1 Exp","G1 Ref","G2 Exp","G2 Ref"),
                                        col=c(cols1[1],cols1[2],cols2[1],cols2[2]))}
        if(etype=="PvT"){
                legend(x="topright",lty=1,cex=0.4,text.col="black", ncol=2,
                                        legend=c("G1 Poly","G1 Total","G2 Poly","G2 Total"),
                                        col=c(cols1[1],cols1[2],cols2[1],cols2[2]))}}}
```

I.2.2.15 sigGenesPlot

```
################
#sigGenesPlot
################
#makes a barplot that shows the # of significant genes

sigGenesPlot<-function(x,toPlot){
        plotNames <- c("e4G1up", "e4G1dn", "e4G2up", "e4G2dn")

        for(i in toPlot){
                eIF4G1up<-length(unlist(x[[i]][1]))
                eIF4G1down<-length(unlist(x[[i]][2]))
                eIF4G2up<-length(unlist(x[[i]][5]))
                eIF4G2down<-length(unlist(x[[i]][6]))

                #plot them
                mp<-barplot(c(eIF4G1up, eIF4G1down, eIF4G2up, eIF4G2down),
                                names.arg=c("eIF4G1up", "eIF4G1down", "eIF4G2up", "eIF4G2down"),
                                col=c("dodgerblue", "dodgerblue4", "darkorange", "darkorange4"),
                                main = names(x)[i], ylab="Number of significant genes", las=3,
                                ylim=c(1, 700), cex.names=0.65)
                text(mp, c(eIF4G1up, eIF4G1down, eIF4G2up, eIF4G2down),
                         labels= c(eIF4G1up, eIF4G1down, eIF4G2up, eIF4G2down), pos=3)}}
```

I.2.2.16 vennDiagrams

```
################
#vennDiagrams
################
#make venn diagrams to compare top lists
#input: list of lists
#arguments: x = list; filename w/o extention, comparisons = vector of list component names of
things to be compared

vennDiagrams<-function(x,comparisons){

#plot the number of significant genes in a topList as a bar plot
        par(mfrow=c(2,2), family = "mono")
        plotNames <- c("e4G1up", "e4G1dn", "e4G2up", "e4G2dn")

        if(length(comparisons)==2){
                sapply(c(1:4),function(i){
                                doVennDiagram(as.character(unlist(x[[comparisons[1]]][i])),

        as.character(unlist(x[[comparisons[2]]][i])),
```

```
        names=c(comparisons[1],comparisons[2]),
                                                                main=plotNames[i])})}
        if(length(comparisons)==3){
                        sapply(c(1:4),function(i){
                                        doVennDiagram(as.character(unlist(x[[comparisons[1]]][i])),

        as.character(unlist(x[[comparisons[2]]][i])),

        as.character(unlist(x[[comparisons[3]]][i])),

        names=c(comparisons[1],comparisons[2],comparisons[3]),
                                                                main=plotNames[i])})}}
```

I.2.2.17 MAplot

```
################
#MAplot
################
#take in fitcontrast object (component of a list)
#and make MA plots for a specific contrast (coef of the fit);
#one with all points, another with sigpts highlighted
#input
        #x = fitcontrast object
        #coef = contrast to be plotted
        #pval = pval cutoff for significance (this is the adj following MT correction)
        #intCO = intensity cutoff (log(2)) above which pts will be colored blue
        #lfcCO = log fold change cutoff above which points will be colored red

MAplot<-function(x, coef, pval,intCO,lfcCO, mName){
        makePlot<-function(x,coef,mtCor,sigpts,pval,intCO, lfcCO){
                plot(x$Amean, x$coefficients[,coef],
                col=rgb(56,56,56,25,maxColorValue=255),
                pch=20,cex=0.35,cex.lab=1,
                xlim=c((min(x$Amean)-0.1*min(x$Amean)),(max(x$Amean)+0.1*max(x$Amean))),
                ylim=c((min(x$coefficients)+0.1*min(x$coefficients)),
                (max(x$coefficients)+0.1*max(x$coefficients))),
                xlab="Amean", ylab=paste("M(",coef,")",sep=""),main=mName)

                if(sigpts){
                        #mark sig pts with an *
                        points(x$Amean[which(mtCor[,"adj.P.Val"]<=pval)],
                                  (x$coefficients[,coef][which(mtCor[,"adj.P.Val"]<=pval)]),
                                       type="p",col=rgb(0,0,0,125,maxColorValue=255), pch=8,
cex=0.5)

                        legend(x="topright",pch=8,cex=0.5,text.col="black",
                           legend=paste("adjPval < ", pval,sep=""),
                           col=rgb(0,0,0,125,maxColorValue=255))}
                abline(0,0,col=rgb(40,40,40,100,maxColorValue=255), lwd=4)
        }#makePlot

        #correct for multiple testing
        mtCor<-topTable(x, coef, n=Inf, genelist=x$genes$ID,
                                adjust.method="BH", sort.by = "none", p.value=1, lfc=0)

        #no point coloring
        par(mfrow=c(2,1), family = "mono")
        makePlot(x,coef,mtCor,sigpts=FALSE,pval=1,intCO=0,lfcCO=0)
        makePlot(x,coef,mtCor,sigpts=TRUE,pval=pval,intCO=intCO,lfcCO=lfcCO)}
```

1.2.2.18 comPlot

```
################
#comPlot
################
#take in fitcontrast object (component of a list)
#and make scatter plots for a specific contrast
#(2 coef of the fit); one with all points,
#another with sigpts highlighted
#input
	#x = fitcontrast object
	#coef = contrast to be plotted
	#pval = pval cutoff for significance (this is the adj following MT correction)
	#intCO = intensity cutoff (log(2)) above which pts will be colored blue
	#lfcCO = log fold change cutoff above which points will be colored red
	#mName = title for plots
comPlot<-function(x, coef1, coef2, pval,intCO,lfcCO, mName){
	makePlot<-function(x,coef1,coef2,mtCor,sigpts,pval,intCO,lfcCO,lim){
		if(lim){
			low=min(x$coefficients)
			high=max(x$coefficients)
			}else{
			low=-2
			high=2}
		plot(x$coefficients[,coef1], x$coefficients[,coef2],
		col=rgb(56,56,56,25,maxColorValue=255),
		pch=20,cex=0.35,cex.lab=1, asp=1,
		xlim=c(low,high),
		ylim=c(low,high),
		xlab=paste("M(",coef1,")",sep=""),
		ylab=paste("M(",coef2,")",sep=""),main=mName)

		if(sigpts){
			#mark sig pts with an *
			points(x$coefficients[,coef1][which(
					mtCor[,"adj.P.Val"]<=pval)],
				x$coefficients[,coef2][which(
					mtCor[,"adj.P.Val"]<=pval)],
				type="p",col=rgb(0,0,0,125,maxColorValue=255),
				pch=8, cex=0.5)
			#color above intensity cutoff as blue
			points(x$coefficients[,coef1][which(
					mtCor[,"adj.P.Val"]<=pval & x$Amean[]>intCO)],
				x$coefficients[,coef2][which(
					mtCor[,"adj.P.Val"]<=pval & x$Amean[]>intCO)],
				type="p",col=rgb(100,149,237,maxColorValue=255),
				pch=8, cex=0.5)
			#color above intensity cutoff AND above log fold change cut off as red
			points(x$coefficients[,coef1][which(
					mtCor[,"adj.P.Val"]<=pval &
					x$Amean[]>intCO &
					(abs(x$coefficients[,coef1])>lfcCO |
					abs(x$coefficients[,coef2])>lfcCO))],
				x$coefficients[,coef2][which(
					mtCor[,"adj.P.Val"]<=pval &
					x$Amean[]>intCO &
					(abs(x$coefficients[,coef1])>lfcCO |
					abs(x$coefficients[,coef2])>lfcCO))],
				type="p",col=rgb(210,105,30,200,maxColorValue=255),
				pch=8, cex=0.5)
```

```
                legend(x="topleft",pch=8,cex=0.5,text.col="black",
                   legend=c(   paste("adjPval < ", pval,sep=""),
                                                paste("Amean > ", intCO, sep=""),
                                                paste("Amean > ", intCO, "+", "LogFC > ",
                                                lfcCO, sep="")),
                   col=c(rgb(0,0,0,125,maxColorValue=255),rgb(100,149,237,
                                                maxColorValue=255), rgb(210,105,30,200,
                                                maxColorValue=255)))}
        abline(0,1,col=rgb(40,40,40,100,maxColorValue=255), lwd=4)}#makePlot

#correct for multiple testing
mtCor<-topTable(x, paste(coef1,"m",coef2,sep=""), n=Inf,
        genelist=x$genes$ID,adjust.method="BH", sort.by = "none",
        p.value=1, lfc=0)

#no point coloring
par(mfcol=c(2,2), family = "mono")
makePlot(x,coef1,coef2,mtCor,sigpts=FALSE,
        pval=1,intCO=0,lfcCO=0,lim=TRUE)
makePlot(x,coef1,coef2,mtCor,sigpts=FALSE,
        pval=1,intCO=0,lfcCO=0,lim=FALSE)
makePlot(x,coef1,coef2,mtCor,sigpts=TRUE,
        pval=pval,intCO=intCO,lfcCO=lfcCO,lim=TRUE)
makePlot(x,coef1,coef2,mtCor,sigpts=TRUE,
        pval=pval,intCO=intCO,lfcCO=lfcCO,lim=FALSE)}
```

I.2.2.19 makeBarPlot

```
#makes bar plots
#called extensively by other functions

makeBarPlot<-function(data, errors, ylim, cols, ylab, xlab, names, main, subtitle){
        library(Hmisc)
        bp<-barplot(   height=data, space=c(0.1,0.75), names.arg = names,
legend.text=rownames(data),
                                        beside=TRUE, horiz=FALSE, xlab= xlab, ylab= ylab,
cex.names=0.5,
                                        ylim=ylim, col=cols, border = "black", main=main)
                mtext(subtitle, side = 3, cex = 0.75, las = 0, padj = -1)
                if(length(errors) > 0){
                errbar(x=bp, y=data, yplus=data+errors, yminus=data-errors,add=TRUE)}

        minor.tick(nx=1, ny=2,tick.ratio=0.5)

        abline(h = 0, lwd=2)
        return(bp)}
```

I.3 Statistical analysis of message characteristics

A number of statistical tests were performed to determine the significance of differences in protein and mRNA abundance, protein decay rate, ORF length and 5'-UTR length. The code used to perform such tests and make representative plots (see Figure 3.8) is presented here.

```
dir<-#USER SPECIFIES LOCATION OF DATA AND CODE

###############
##read in and format the data sets:
###############
```

```
#individual mRNA characteristic data  sets
##von Der Haar
	vonDerHaarData<-read.csv(file=paste(dir,"vonDerHaarCurated.csv", sep=""),
		colClasses=c(rep("character",2),rep("numeric",5)))
	names(vonDerHaarData)<-c("ID","Name","MW","Length","ProteinAbundance",
		"ProteinDecay","mRNAAbundance")
	save(vonDerHaarData, file=paste(dir, "vonDerHaarData.Rdata", sep=""))

##yassour
	yassourSD2<-read.csv(file=paste(dir,"yassourPNAS2009TableSD2.csv", sep=""),
		colClasses=c(rep("character",2),rep("numeric",5)))
	names(yassourSD2)<-c("ID","Name","Chr","TSS","5AUG","3STOP","END")

	#calculate the UTR lengths
		yassourSD2[,"5UTR"]<-abs(yassourSD2[,"TSS"]-yassourSD2["5AUG"])
		yassourSD2[,"3UTR"]<-abs(yassourSD2[,"END"]-yassourSD2["3STOP"])

	#fix the data points with non-numerical values
	for(i in 1:dim(yassourSD2)[2]){
		yassourSD2[,i][which(yassourSD2[,i]=="NaN")]<-NA
		yassourSD2[,i][which(yassourSD2[,i]=="Inf")]<-NA}

	save(yassourSD2, file=paste(dir,"yassourSD2.Rdata", sep=""))

##nagalakshmi
	nagalakshmiS4<-read.csv(file=paste(dir,"NagalakshmiScience2008TableS4.csv",
		sep=""), colClasses=c(rep("character",2),rep("numeric",4),
		rep("character",1),rep("numeric",2)))
	names(nagalakshmiS4)[1]<-"ID"
	names(nagalakshmiS4)[7]<-"5UTR"
	names(nagalakshmiS4)[8]<-"3UTR"
	names(nagalakshmiS4)[9]<-"TNXlevel"

	#clean up non-numerics
		nagalakshmiS4[,"5UTR"][which(
			nagalakshmiS4[,"5UTR"]=="Potential_AUG_annotation_error")]<-NA
		nagalakshmiS4[,"5UTR"]<-as.numeric(nagalakshmiS4[,"5UTR"])
		for(i in 1:dim(nagalakshmiS4)[2]){
			nagalakshmiS4[,i][which(nagalakshmiS4[,i]=="NaN")]<-NA
			nagalakshmiS4[,i][which(nagalakshmiS4[,i]=="Inf")]<-NA}

	#calculate ORF length
	nagalakshmiS4[,"ORFlength"]<-abs(
		nagalakshmiS4[,"SGD_Start"]-nagalakshmiS4[,"SGD_End"])

	save(nagalakshmiS4, file=paste(dir,"nagalakshmiS4.Rdata", sep=""))

##muira
	miuraS4<-read.csv(file=paste(dir,"miuraPNAS2006TableS4.csv", sep=""),
		colClasses=c(rep("character",5),rep("numeric",2),rep("character",1),
		rep("numeric",2),rep("character",1)))
	names(miuraS4)[2]<-"ID"
	names(miuraS4)[6]<-"5UTR"
	names(miuraS4)[7]<-"NumberOfuATG"
	names(miuraS4)[9]<-"LongestuORF"
	names(miuraS4)[10]<-"ShortestuORF"

	#fix non-numeric 5' utrs
	for(i in 1:dim(miuraS4)[2]){
		miuraS4[,i][which(miuraS4[,i]=="NaN")]<-NA
```

```
                miuraS4[,i][which(miuraS4[,i]=="Inf")]<-NA}

        #This object has a number of different 5' UTR lengths for each ORF and
        #they need to be combinded. Basically I want to take the mean
        #make an object where all the stats for each ORF are combined

        #create an object that has as many rows are there are
        #unique entries in the Miura list
        miuraStats<-as.data.frame(matrix(NA,nrow=length(
                        unique(miuraS4[,"ID"])),ncol=4))

        #populate the object with means and medians for each entry
        for(i in 1:length(unique(miuraS4[,"ID"]))){
                miuraStats[i,1]<-unique(miuraS4[,"ID"])[i]
                miuraStats[i,2]<-mean(unlist(miuraS4[,"5UTR"][which(
                                unique(miuraS4[,"ID"])[i]==miuraS4[,"ID"])]),
                                na.rm=TRUE)
                miuraStats[i,3]<-median(unlist(miuraS4[,"5UTR"][which(
                                unique(miuraS4[,"ID"])[i]==miuraS4[,"ID"])]),
                                na.rm=TRUE)
                miuraStats[i,4]<-max(unlist(miuraS4[,"NumberOfuATG"][which(
                                unique(miuraS4[,"ID"])[i]==miuraS4[,"ID"])]),
                                na.rm=TRUE)}
        names(miuraStats)<-c("ID","means","medians","uORFs")
        save(miuraStats, file=paste(dir,"miuraStats.Rdata", sep=""))

#my microarray data
        dir<-"/Users/Bryan/Desktop"
        load(file=paste(dir, "/E12.1.dataList.Rdata", sep=""))

        load(file=paste(dir,"/vonDerHaarData.Rdata", sep=""))
        load(file=paste(dir,"/yassourSD2.Rdata", sep=""))
        load(file=paste(dir,"/nagalakshmiS4.Rdata", sep=""))
        load(file=paste(dir,"/miuraStats.Rdata", sep=""))

        load(file=paste(dir,"expressed.Rdata", sep=""))
        load(file=paste(dir,"enhanced.Rdata", sep=""))
        load(file=paste(dir,"repressed.Rdata", sep=""))


###############
##organize the data into a single matrix
###############
dataNames<-c("ID","ORFlength","ProteinAbundance","ProteinDecayRate",
        "mRNAabundance","yassour","nagalakshmi","miuraMeans","miuraMedians")

#create matricies filled with NA to be populated with UTR data
#one for each expressed, PvTenhanced, and PvTrepressed
        expressed<-as.data.frame(matrix(NA,nrow=length(which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)),
                ncol=length(dataNames)))
        enhanced<-as.data.frame(matrix(NA,nrow=length(which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)),
                ncol=length(dataNames)))
        repressed<-as.data.frame(matrix(NA,nrow=length(which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
```

```
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)),
                ncol=length(dataNames)))

#populate
#expressed
        for(i in 1:dim(expressed)[1]){
        expressed[i,1]<-dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]]
        if(length(which(
        vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]]))>0){
        expressed[i,2]<-vonDerHaarData[,"Length"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                        dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]
        expressed[i,3]<-vonDerHaarData[,"ProteinAbundance"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                        dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]
        expressed[i,4]<-vonDerHaarData[,"ProteinDecay"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                        dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]
        expressed[i,5]<-vonDerHaarData[,"mRNAAbundance"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                        dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]}else{
                expressed[i,2]<-NA
                expressed[i,3]<-NA
                expressed[i,4]<-NA
                expressed[i,4]<-NA
                expressed[i,5]<-NA}
        if(length(which(
        yassourSD2[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]]))>0){
                        expressed[i,6]<-yassourSD2[,"5UTR"][which(
                yassourSD2[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]}
        else{expressed[i,6]<-NA}

        if(length(which(
        nagalakshmiS4[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]]))>0){
                        expressed[i,7]<-nagalakshmiS4[,"5UTR"][which(
        nagalakshmiS4[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]}
        else{expressed[i,7]<-NA}

        if(length(which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]]))>0){
```

```
                expressed[i,8]<-miuraStats[,"means"][which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]
        else{expressed[i,8]<-NA}
        if(length(which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]]))>0){
                expressed[i,9]<-miuraStats[,"medians"][which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3)[i]])]
        else{expressed[i,9]<-NA}}
        colnames(expressed)<-dataNames

#enhanced
        for(i in 1:dim(enhanced)[1]){
        enhanced[i,1]<-dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]
        if(length(which(
        vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])) > 0){
        enhanced[i,2]<-vonDerHaarData[,"Length"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]
        enhanced[i,3]<-vonDerHaarData[,"ProteinAbundance"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]
        enhanced[i,4]<-vonDerHaarData[,"ProteinDecay"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]
        enhanced[i,5]<-vonDerHaarData[,"mRNAAbundance"][which(
                vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{
                enhanced[i,2]<-NA
                enhanced[i,3]<-NA
                enhanced[i,4]<-NA
                enhanced[i,4]<-NA
                enhanced[i,5]<-NA}
```

```
        if(length(which(
        yassourSD2[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                        enhanced[i,6]<-yassourSD2[,"5UTR"][which(
        yassourSD2[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{enhanced[i,6]<-NA}

        if(length(which(
        nagalakshmiS4[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                        enhanced[i,7]<-nagalakshmiS4[,"5UTR"][which(
        nagalakshmiS4[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{enhanced[i,7]<-NA}

        if(length(which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                        enhanced[i,8]<-miuraStats[,"means"][which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{enhanced[i,8]<-NA}
        if(length(which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                        enhanced[i,9]<-miuraStats[,"medians"][which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]>0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{enhanced[i,9]<-NA}}
        colnames(enhanced)<-dataNames

#repressed
        for(i in 1:dim(repressed)[1]){
        repressed[i,1]<-dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
```

```
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]
if(length(which(
vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])) > 0){
repressed[i,2]<-vonDerHaarData[,"Length"][which(
        vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]
repressed[i,3]<-vonDerHaarData[,"ProteinAbundance"][which(
        vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]
repressed[i,4]<-vonDerHaarData[,"ProteinDecay"][which(
        vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]
repressed[i,5]<-vonDerHaarData[,"mRNAAbundance"][which(
        vonDerHaarData[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
else{
        repressed[i,2]<-NA
        repressed[i,3]<-NA
        repressed[i,4]<-NA
        repressed[i,4]<-NA
        repressed[i,5]<-NA}
if(length(which(
yassourSD2[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                repressed[i,6]<-yassourSD2[,"5UTR"][which(
yassourSD2[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
else{repressed[i,6]<-NA}
if(length(which(
nagalakshmiS4[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
        dataList[[2]][["AquantScale"]]$Amean>7.3 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                repressed[i,7]<-nagalakshmiS4[,"5UTR"][which(
nagalakshmiS4[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
```

```
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{repressed[i,7]<-NA}
        if(length(which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                        repressed[i,8]<-miuraStats[,"means"][which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{repressed[i,8]<-NA}
        if(length(which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]]))>0){
                        repressed[i,9]<-miuraStats[,"medians"][which(
        miuraStats[,"ID"]==dataList[[2]][["AquantScale"]]$genes$ID[which(
                dataList[[2]][["AquantScale"]]$genes$Type != "OCH" &
                dataList[[2]][["AquantScale"]]$Amean>7.3 &
                dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"]<0 &
                dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"]<0.05)[i]])]}
        else{repressed[i,9]<-NA}}
        colnames(repressed)<-dataNames

        compiledFeatureCharacteristics<-list(expressed,enhanced,repressed)
        names(compiledFeatureCharacteristics)<-c("expressed","enhanced","repressed")
        save(compiledFeatureCharacteristics,file=paste(dir,
                "compiledFeatureCharacteristics.Rdata", sep=""))

###############
##plots
###############
        dataNames<-c("ID","ORFlength","ProteinAbundance","ProteinDecayRate",
                "mRNAabundance","yassour","nagalakshmi","miuraMeans","miuraMedians")

pdf(paste(dir,"noOCHprocessingPlots.pdf"))
        par(mfrow=c(2,2))

        #von der haar
        xAxis<-list(c(-400,5000,0,800000,-0.025,0.4,0,80),
                c(-400,5000,-2500,40000,-0.025,0.4,0,15))
        yAxis<-c(0,0.004,0,0.0004,0,55,0,1.0)
        for(j in 2:5){
        for(i in 1:2){
        plot(density(compiledFeatureCharacteristics[["expressed"]][,j],na.rm=TRUE),
                main=paste("Von Der Haar ", dataNames[j],sep=""), xlab= dataNames[j],
                col="gray20",lwd=2,
                xlim=c(xAxis[[i]][seq(from=1,to=7,by=2)[j-1]],
                xAxis[[i]][seq(from=2,to=8,by=2)[j-1]]),
                ylim=c(yAxis[seq(from=1,to=7,by=2)[j-1]],
                yAxis[seq(from=2,to=8,by=2)[j-1]]))
```

```
        lines(density(compiledFeatureCharacteristics[["enhanced"]][,j],na.rm=TRUE),
                col="firebrick",lwd=2)
        lines(density(compiledFeatureCharacteristics[["repressed"]][,j],na.rm=TRUE),
                col="chartreuse4",lwd=2)
        legend(x="topright", legend=c("Expressed", "Enhanced", "Repressed"),
                col=c("gray20","firebrick","chartreuse4"), lty=1, cex=0.75)}}

#5'UTR stuff
        par(mfrow=c(3,2))
        #all expressed genes from the different data sets; to compare how similar they are
        plot(density(compiledFeatureCharacteristics[[
                "expressed"]][,"yassour"],na.rm=TRUE),
                col="gray20", lwd=2, ylim=c(0,0.015), xlim=c(0,1000),
                main="5' UTR Distributions for Expressed Genes",
                xlab="5' UTR Length", ylab="Density")
        lines( density(compiledFeatureCharacteristics[[
                "expressed"]][,"nagalakshmi"],na.rm=TRUE), col="gray40", lwd=2)
        lines( density(compiledFeatureCharacteristics[[
                "expressed"]][,"miuraMeans"],na.rm=TRUE), col="gray60", lwd=2)
        lines( density(compiledFeatureCharacteristics[[
                "expressed"]][,"miuraMedians"],na.rm=TRUE), col="gray80", lwd=2)
        legend(x="topright", legend=c("yassour","nagalakshmi","miuraMeans",
                "miuraMedians"), col=c("gray20","gray40","gray60","gray80"),
                lty=1, cex=0.75)
        plot.new()

        for(i in 6:9){
        plot(   density(compiledFeatureCharacteristics[[
                "expressed"]][,i],na.rm=TRUE), col="gray40", lwd=2,
                main=paste(dataNames[i], " 5' UTR Distributions", sep=""),
                ylim=c(0,0.015), xlim=c(0,1000), xlab="5' UTR Length",
                ylab="Density")
        lines( density(compiledFeatureCharacteristics[[
                "enhanced"]][,i],na.rm=TRUE), col="firebrick", lwd=2)
        lines( density(compiledFeatureCharacteristics[[
                "repressed"]][,i],na.rm=TRUE), col="chartreuse4", lwd=2)
        legend(x="topright",
                legend=c(      "expressed","enhanced", "repressed"),
#               legend=c(paste(       "expressed (", format(mean(expressed[,i],
#                       na.rm=TRUE),digits=1), ";",
#                       format(median(expressed[,i], na.rm=TRUE),digits=1),";n=",
#                       length(which(!is.na(expressed[,i]))),")",sep=""),
#                       paste(  "enhanced (", format(mean(enhanced[,i],
#                       na.rm=TRUE),digits=1), ";",
#                       format(median(enhanced[,i], na.rm=TRUE),digits=1),";n=",
#                       length(which(!is.na(enhanced[,i]))),")",sep=""),
#                       paste(  "repressed (", format(mean(repressed[,i],
#                       na.rm=TRUE),digits=1), ";",
#                       format(median(repressed[,i], na.rm=TRUE),digits=1),";n=",
#                       length(which(!is.na(repressed[,i]))),")",sep="")),
                col=c("gray40","firebrick","chartreuse4"), lty=1, cex=0.75)}
dev.off()

###############
##stats
###############
        statNames<-c("dataset","n(exp)","expMean","expMed","n(enh)",
                                "enhMean","enhMed", "t.test","wilcox","n(rep)",
                                "repMean","repMed","t.test","wilcox")
```

```
        statTests<-as.data.frame(matrix(NA,nrow=(length(dataNames)-1),
                ncol=length(statNames)))
        statTests[,1]<-dataNames[2:length(dataNames)]
        colnames(statTests)<-statNames

        for(i in 2:9){
        statTests[i-1,2]<-format(length(which(
                !is.na(compiledFeatureCharacteristics[["expressed"]][,i]))),
                digits=1)
        statTests[i-1,3]<-format(mean(
                compiledFeatureCharacteristics[["expressed"]][,i],
                na.rm=TRUE), digits=1)
        statTests[i-1,4]<-format(median(
                compiledFeatureCharacteristics[["expressed"]][,i],
                na.rm=TRUE), digits=1)

        statTests[i-1,5]<-format(length(which(
                !is.na(compiledFeatureCharacteristics[["enhanced"]][,i]))),
                digits=1)
        statTests[i-1,6]<-format(mean(
                compiledFeatureCharacteristics[["enhanced"]][,i],
                na.rm=TRUE),digits=1)
        statTests[i-1,7]<-format(median(
                compiledFeatureCharacteristics[["enhanced"]][,i],
                na.rm=TRUE),digits=1)
        statTests[i-1,8]<-format(t.test(
                compiledFeatureCharacteristics[["expressed"]][,i],
                compiledFeatureCharacteristics[["enhanced"]][,i])$"p.value",
                digits=3)
        statTests[i-1,9]<-format(wilcox.test(
                compiledFeatureCharacteristics[["expressed"]][,i],
                compiledFeatureCharacteristics[["enhanced"]][,i])$"p.value",
                digits=3)

        statTests[i-1,10]<-format(length(which(
                !is.na(compiledFeatureCharacteristics[["repressed"]][,i]))),
                digits=1)
        statTests[i-1,11]<-format(mean(
                compiledFeatureCharacteristics[["repressed"]][,i],
                na.rm=TRUE),digits=1)
        statTests[i-1,12]<-format(median(
                compiledFeatureCharacteristics[["repressed"]][,i],
                na.rm=TRUE),digits=1)
        statTests[i-1,13]<-format(t.test(
                compiledFeatureCharacteristics[["expressed"]][,i],
                compiledFeatureCharacteristics[["repressed"]][,i])$"p.value",
                digits=3)
        statTests[i-1,14]<-format(wilcox.test(
                compiledFeatureCharacteristics[["expressed"]][,i],
                compiledFeatureCharacteristics[["repressed"]][,i])$"p.value",
                digits=3)}

#               cbind(statTests[,4],statTests[,2],statTests[,7],statTests[,5],
#               statTests[,9],statTests[,12],statTests[,10],statTests[,14])

#PASTA lists
        PASTAlist<-read.csv(file=paste(dir,"PASTALongAndShort.csv", sep=""),
                colClasses="character")

        justORFs<-matrix(unlist(lapply(list(
```

```
expressedOverlap<-intersect(dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF")],
        PASTAlist[,"AllNames"]),
expressedShort<-intersect(dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF")],
        PASTAlist[,"ShortID"]),
expressedLong<-intersect(dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF")],
        PASTAlist[,"LongID"]),

enhancedOverlap<-intersect(dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] > 0)],
        PASTAlist[, "AllNames"]),
enhancedShort<-intersect(     dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] > 0)],
        PASTAlist[, "ShortID"]),
enhancedLong<-intersect(      dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] > 0)],
        PASTAlist[, "LongID"]),

repressedOverlap<-intersect(  dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] < 0)],
        PASTAlist[, "AllNames"]),
repressedShort<-intersect(    dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] < 0)],
        PASTAlist[, "ShortID"]),
repressedLong<-intersect(     dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] < 0)],
        PASTAlist[, "LongID"])),length)),nrow=3,ncol=3,byrow=TRUE)
colnames(justORFs)<-c("overlap","short","long")
rownames(justORFs)<-c("expressed","enhanced","repressed")
justORFs

#          overlap short long
#expressed    4980   480  701
#enhanced      133    15    5
#repressed     127     3   32

#fisher tests
```

```
        enhancedVshort<-
                matrix(c(
                length(enhancedShort),
                length(setdiff(enhancedOverlap,enhancedShort)),
                length(setdiff(expressedShort,enhancedOverlap)),
                length(setdiff(setdiff(expressedOverlap,expressedShort),
                enhancedOverlap))),ncol=2,nrow=2,byrow=TRUE)
                #     [,1] [,2]
                #[1,]   15  118
                #[2,]  465 4382
        #15 out of 133 enhanced genes have short polyA tails = 11%
        #480 out of 4980 expressed genes have short polyA tails = 9.6%

        fisher.test(enhancedVshort)
                #Fisher's Exact Test for Count Data
                #data:  enhancedVshort
                #p-value = 0.5498
                #alternative hypothesis: true odds ratio is not equal to 1
                #95 percent confidence interval:
                # 0.6443134 2.0788638
                #sample estimates:
                #odds ratio
                #  1.197875

enhancedVlong<-
        matrix(c(
        length(enhancedLong),
        length(setdiff(enhancedOverlap,enhancedLong)),
        length(setdiff(expressedLong,enhancedOverlap)),
        length(setdiff(setdiff(expressedOverlap,expressedLong),
                enhancedOverlap))),ncol=2,nrow=2,byrow=TRUE)
        #     [,1] [,2]
        #[1,]    5  128
        #[2,]  696 4152
#5 of 133 enhanced genes have a long pA tail (3.7%)
#701 of 4980 expressed genes have a long pA tail (14%)

fisher.test(enhancedVlong)
        #Fisher's Exact Test for Count Data
        #data:  enhancedVlong
        #p-value = 0.0001218
        #alternative hypothesis: true odds ratio is not equal to 1
        #95 percent confidence interval:
        # 0.07414056 0.56119986
        #sample estimates:
        #odds ratio
        # 0.2330687

repressedVshort<-
matrix(c(
        length(repressedShort),
        length(setdiff(repressedOverlap,repressedShort)),
        length(setdiff(expressedShort,repressedOverlap)),
        length(setdiff(setdiff(expressedOverlap,expressedShort),
                repressedOverlap))),ncol=2,nrow=2,byrow=TRUE)

        #     [,1] [,2]
        #[1,]    3  124
        #[2,]  477 4376
#3 out of 127 repressed genes have short polyA tails = 2.4%
```

```
#480 out of 4980 expressed genes have short polyA tails = 9.6%

fisher.test(repressedVshort)
        #Fisher's Exact Test for Count Data
        #data:  repressedVshort
        #p-value = 0.002062
        #alternative hypothesis: true odds ratio is not equal to 1
        #95 percent confidence interval:
        # 0.04499506 0.66830745
        #sample estimates:
        #odds ratio
        # 0.2219899

repressedVlong<-
matrix(c(
        length(repressedLong),
        length(setdiff(repressedOverlap,repressedLong)),
        length(setdiff(expressedLong,repressedOverlap)),
        length(setdiff(setdiff(expressedOverlap,expressedLong),
                repressedOverlap))),ncol=2,nrow=2,byrow=TRUE)
        #     [,1] [,2]
        #[1,]   32  95
        #[2,]  668 4181
#32 of 127 repressed genes have a long pA tail (25%)
#701 of 4980 expressed genes have a long pA tail (14%)

fisher.test(repressedVlong)
        #Fisher's Exact Test for Count Data
        #data:  repressedVlong
        #p-value = 0.0006799
        #alternative hypothesis: true odds ratio is not equal to 1
        #95 percent confidence interval:
        # 1.353626 3.204939
        #sample estimates:
        #odds ratio
        #  2.106765

##summary
#          overlap short long
#expressed    4980   480  701
#enhanced      133    15    5
#repressed     127     3   32

#expressed
#480 out of 4980 expressed ORFs have short polyA tails = 9.6%
#701 of 4980 expressed ORFs have a long pA tail (14%)

#ehanced
#15 out of 133 enhanced ORFs have short polyA tails = 11%; p-value = 0.5498
#5 of 133 enhanced ORFs have a long pA tail = 3.7%; p-value = 0.0001218

#repressed
#3 out of 127 repressed ORFs have short polyA tails = 2.4%; p-value = 0.002062
#32 of 127 repressed ORFs have a long pA tail (25%); p-value = 0.0006799


#now need to recalc the pab associations with just the ORFs
pabData<-read.csv(file=paste(dir,"hoganS3pab.csv", sep=""),
        colClasses=c(rep("character",1),rep("numeric",6)))
        names(pabData)[1]<-"GeneID"
```

```
        names(pabData)[6]<-"FDR"
        names(pabData)[7]<-"LocalFDR"

#hogan data has the stanford names, which need to be converted;
probeNameConverter<-function(x){
        yOligoConverter<-read.csv(paste(dir,"yOligoNameConverter.csv", sep=""),
                        colClasses="character")
        yBOXConverter<-read.csv(paste(dir,"yBoxNameConverter.csv", sep=""),
                        colClasses="character")

        if(length(x$GeneID)>0){
                for(y in 1:length(x$GeneID)){
                        if(length(which(yOligoConverter$StanName==x$GeneID[y]))>0){
                                x$GeneID[y]<-yOligoConverter$ID[which(
                                        yOligoConverter$StanName==x$GeneID[y])]}
                        if(length(which(yBOXConverter$ID==x$GeneID[y]))>0){
                                x$GeneID[y]<-yBOXConverter$standardName[which(
                                        yBOXConverter$ID==x$GeneID[y])][1]}
                        firstFour<-substr(x$GeneID[y],start=1,stop=4)
                        lastFour<-substr(x$GeneID[y],start=(nchar(x$GeneID[y])-3),
                        stop=nchar(x$GeneID[y]))

                        if(firstFour == "ARS_" | firstFour == "CEN_" | firstFour == "RRN_"
|
                                firstFour == "TEL_" | firstFour == "ARS_"){
                                        x$GeneID[y]<-sub("^.{4}","",x$GeneID[y])}
                        if(lastFour == "_ARS" | lastFour == "_CEN" | lastFour == "_NCR"){
                                        x$GeneID[y]<-sub(".{4}$","",x$GeneID[y])}
                        if(substr(x$GeneID[y],start=(nchar(x$GeneID[y])-4),
                                stop=nchar(x$GeneID[y])) == "_RRNA"){
                        if(substr(x$GeneID[y],start=1,stop=1) == "E" |
                                substr(x$GeneID[y],start=1,stop=1) == "I" |
                                substr(x$GeneID[y],start=1,stop=1) == "N" |
                                substr(x$GeneID[y],start=1,stop=1) == "R"){
                                        x$GeneID[y]<-sub(".{5}$","",x$GeneID[y])}}
                        if(substr(x$GeneID[y],start=1,stop=3)=="ARS"){
                                        x$GeneID[y]<-sub("_.*$","",x$GeneID[y])
                                        x$GeneID[y]<-sub("-.*$","",x$GeneID[y])}
                        if(substr(x$GeneID[y],start=(nchar(x$GeneID[y])-6),
                                stop=nchar(x$GeneID[y])) == "_SNORNA"){
                                        x$GeneID[y]<-sub(".{7}$","",x$GeneID[y])}
                        if(substr(x$GeneID[y],start=(nchar(x$GeneID[y])-5),
                                stop=nchar(x$GeneID[y])) == "_SNRNA"){
                                        x$GeneID[y]<-sub(".{6}$","",x$GeneID[y])}
                        if(substr(x$GeneID[y],start=(nchar(x$GeneID[y])-4),
                                stop=nchar(x$GeneID[y])) == "_TRNA"){
                                        x$GeneID[y]<-sub(".{5}$","",x$GeneID[y])}
                        if(firstFour == "OCH_"){#to deal with the OCH named probes
                                if(     substr(x$GeneID[y],start=1,stop=10)=="OCH_DS100_" |
                                        substr(x$GeneID[y],start=1,stop=10)=="OCH_US100_" ){
                                        x$GeneID[y]<-sub("OCH_.S100_","",x$GeneID[y])
                                }else{#for the "OCH_systematic_common"
                                        x$GeneID[y]<-sub("OCH_","",x$GeneID[y])
                                        x$GeneID[y]<-sub("_.*","",x$GeneID[y])}}
                        if(substr(x$GeneID[y],start=1,stop=3) == "CEN"){
                                x$GeneID[y]<-sub("_.*$","",x$GeneID[y])}}
                return(x)}}

pabData<-probeNameConverter(pabData)
```

```
justORFsPab<-matrix(unlist(lapply(list(
expressedPabOverlap<-intersect(dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF")],
        pabData[,"GeneID"]),
expressedPab<-intersect(dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF")],
        pabData[,"GeneID"][which(pabData$FDR<1)]),

enhancedPabOverlap<-intersect(       dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] > 0)],
        pabData[,"GeneID"]),
enhancedPab<-intersect(       dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] > 0)],
        pabData[,"GeneID"][which(pabData$FDR<1)]),

repressedPabOverlap<-intersect(      dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] < 0)],
        pabData[,"GeneID"]),
repressedPab<-intersect(      dataList[[2]][["AquantScale"]]$genes$ID[which(
        dataList[[2]][["AquantScale"]]$Amean > 7.3 &
        dataList[[2]][["AquantScale"]]$genes$Type == "ORF" &
        dataList[[2]][["AquantScale"]]$adj.P.Val[,"e4G1mWT"] < 0.05 &
        dataList[[2]][["AquantScale"]]$coefficients[,"e4G1mWT"] < 0)],
        pabData[,"GeneID"][which(pabData$FDR<1)])),length)),nrow=3,ncol=2,byrow=TRUE)
colnames(justORFsPab)<-c("overlap","pab")
rownames(justORFsPab)<-c("expressed","enhanced","repressed")
justORFsPab

#          overlap  pab
#expressed    4040 1948
#enhanced      118   55
#repressed     113   73

enhancedVpab<-
        matrix(c(
        length(enhancedPab),
        length(setdiff(enhancedPabOverlap,enhancedPab)),
        length(setdiff(expressedPab,enhancedPabOverlap)),

length(setdiff(setdiff(expressedOverlap,expressedPab),
        enhancedPabOverlap))),ncol=2,nrow=2,byrow=TRUE)

        #     [,1] [,2]
        #[1,]   55   63
        #[2,] 1893 3082
#55 out of 118 enhanced genes are associated with pab = 46%
#1948 out of 4040 expressed genes are associated with pab = 48%

fisher.test(enhancedVpab)
```

```
        #Fisher's Exact Test for Count Data
        #data:  enhancedVpab
        #p-value = 0.06825
        #alternative hypothesis: true odds ratio is not equal to 1
        #95 percent confidence interval:
        # 0.966889 2.083675
        #sample estimates:
        #odds ratio
        #  1.421218

repressedVpab<-
        matrix(c(
        length(repressedPab),
        length(setdiff(repressedPabOverlap,repressedPab)),
        length(setdiff(expressedPab,repressedPabOverlap)),

length(setdiff(setdiff(expressedOverlap,expressedPab),
        repressedPabOverlap))),ncol=2,nrow=2,byrow=TRUE)
        #     [,1] [,2]
        #[1,]   73   40
        #[2,] 1875 3103
#73 of 113 repressed genes have a long pA tail (64%)
#1948 out of 4040 expressed genes are associated with pab = 48%

fisher.test(repressedVpab)
        #Fisher's Exact Test for Count Data
        #data:  repressedVpab
        #
        #p-value = 1.194e-08
        #alternative hypothesis: true odds ratio is not equal to 1
        #95 percent confidence interval:
        # 2.016853 4.578174
        #sample estimates:
        #odds ratio
        #  3.019575

##summary
        #          overlap  pab
        #expressed    4040 1948
        #enhanced      118   55
        #repressed     113   73
#1948 out of 4040 expressed genes are associated with pab = 48.2%
#55 of 118 enhanced genes are associated with pab = 46.6% (p-value = 0.06825)
#73 of 113 repressed genes are asscociated with pab = 64.4% (p-value = 1.194e-08)
```

I.4 *In vitro* translation data processing

To determine the reproducibility of findings, numerous *in vitro* translation experiments examining the relationship between poly(A) tail length and translational efficiency were performed. These experiments all had a similar design, so a function to process and compile the results was written and presented here. As with the above sections the code calling to function is presented first to provide a conceptual understanding of what is being computed. This is then followed by detailed code containing the function that performs most of the computational work.

I.4.1 High Level Code

```
library(gdata)
dir<-#USER SPECIFIES LOCATION OF DATA
pATailLength<-c(sort(c(seq(from = 0, to = 100, by = 10), 15), decreasing = TRUE))

	#HiRespAIVT experiments separate
	subtitle = FALSE
	RNAinput<-c(120)
	ylimRaw<-list(c(0, 10000), c(0, 5000), c(0, 3000))
	ylimNorm<-c(0,1.25)
	filenames<-lapply(21:23, function(x){
		c(paste("HiRespAIVT", x, "wt", 1:4, sep = ""), paste("HiRespAIVT", x, "d4G1e",
1:4, sep = ""))})
	identifier<-lapply(1:length(filenames), function(x){c(rep("WT", times = 4), rep("d4G1",
times = 4))})
	comparisons<-lapply(1:length(filenames), function(x){c("WT","d4G1","d4G1","WT")})
	source(paste(dir, "code/processIVTcomparisons.R",sep=""))
	lapply(1:length(filenames), function(i){
		processIVTcomparisons(filenames[[i]], RNAinput[i], pATailLength, ylimNorm,
ylimRaw[[i]],
			subtitle, identifier[[i]], comparisons[[i]])})

	#HiRespAIVT experiments combined
	subtitle = FALSE
	RNAinput<-c(120)
	ylimRaw<-c(0, 10000)
	ylimNorm<-c(0,1.25)
	filenames<-unmatrix(sapply(21:23, function(x){
		c(paste("HiRespAIVT", x, "wt", 1:4, sep = ""), paste("HiRespAIVT", x, "d4G1e",
1:4, sep = ""))}))
	identifier<-unmatrix(sapply(1:(length(filenames)/8), function(x){c(rep("WT", times = 4),
rep("d4G1", times = 4))}))
	comparisons<-c("WT","d4G1","d4G1","WT")
	source(paste(dir, "code/processIVTcomparisons.R",sep=""))
	processIVTcomparisons(filenames, RNAinput, pATailLength, ylimNorm, ylimRaw,
		subtitle, identifier, comparisons)
```

I.4.2 Functions

```
#combine similar and compare different
#in vitro translation experiments

processIVTcomparisons<-function(filenames, RNAinput, pATailLength,
	ylimNorm, ylimRaw, subtitle, identifier, comparisons){

####################
##read in the data
####################
	library(Hmisc)

	dataList<-lapply(filenames,function(i){
		read.csv(file = paste(dir, "data/", i, ".csv", sep=""),
			stringsAsFactors = FALSE)})
	names(dataList)<-filenames

#reverse order so data is from short->long pA length
	for(x in 1:length(dataList)){
		for(i in 1:dim(dataList[[x]])[2]){
			dataList[[x]][,i]<-dataList[[x]][nrow(dataList[[x]]):1,i]}}
	pATailLength<-rev(pATailLength)
####################
```

```
##process and normalize
####################
#percent error for within experiment replicates
        for(x in 1:length(dataList)){
        for(i in 1:dim(dataList[[x]])[1]){
                for(j in 3:(3+length(RNAinput)-1)){
                        dataList[[x]][i,(j+2*length(RNAinput))]<-(dataList[[x]][
                                i,(j+length(RNAinput))]/dataList[[x]][i,j])}}
                        names(dataList[[x]])[(j+2*length(RNAinput))]<-paste(
                                "percentError", RNAinput[j-2], sep="")}
        write.table(dataList[[1]], file = paste(dir,"plots/rData/",
                filenames[1],"_RawData.tab", sep=""), sep = "\t")

#normalize to max input (visualize concentration dependence);
#keep inputs and experiments separate
        if(length(RNAinput) > 1){
        inputNormList<-dataList
                for(x in 1:length(inputNormList)){
                for(i in 1:dim(inputNormList[[x]])[1]){
                        for(j in 3:(3+length(RNAinput)-1)){
                        #norm the data
                        inputNormList[[x]][i,j]<-(dataList[[x]][i,j]/
                                dataList[[x]][i,(2+length(RNAinput))])
                        #norm the errors
                        inputNormList[[x]][i,(j+length(RNAinput))]<-
                                inputNormList[[x]][i,j]*sqrt(
                                (dataList[[x]][i,(2+2*length(RNAinput))]/
                                        dataList[[x]][i,(2+length(RNAinput))])^2 +
                                (dataList[[x]][i,(j+length(RNAinput))]/
                                        dataList[[x]][i,j])^2)
                        inputNormList[[x]][i,(j+2*length(RNAinput))]<-
                                (inputNormList[[x]][i,(j+length(RNAinput))]/
                                        inputNormList[[x]][i,j])}}
                        names(inputNormList[[x]])[(j+2*length(RNAinput))]<-
                                paste(RNAinput[j-2], "percentError", sep="")}
        RNAinputNorm<-c(sapply(1:length(RNAinput), function(i){
                        as.numeric(RNAinput[i]/max(RNAinput))}))}

#normalize to max pA length (visualize tail length dependence);
#keep inputs and experiments separate
        pANormList<-dataList
                for(x in 1:length(pANormList)){
                for(i in 1:dim(pANormList[[x]])[1]){
                for(j in 3:(3+length(RNAinput)-1)){
                        #norm the data
                        pANormList[[x]][i,j]<-(dataList[[x]][i,j]/
                                dataList[[x]][nrow(dataList[[x]]),j])
                        #norm the errors
                        pANormList[[x]][i,(j+length(RNAinput))]<-
                                pANormList[[x]][i,j]*sqrt(
                                (dataList[[x]][nrow(dataList[[x]]),(j+length(RNAinput))]/
                                        dataList[[x]][nrow(dataList[[x]]),j])^2 +
                                (dataList[[x]][i,(j+length(RNAinput))]/
                                        dataList[[x]][i,j])^2)
                        pANormList[[x]][i,(j+2*length(RNAinput))]<-(pANormList[[x]][
                                i,(j+length(RNAinput))]/pANormList[[x]][i,j])}}
                        names(pANormList[[x]])[(j+2*length(RNAinput))]<-
                                paste(RNAinput[j-2],"percentError",sep="")}
        pALengthNorm<-c(sapply(1:length(pATailLength), function(i){
                as.numeric(pATailLength[i]/max(pATailLength))}))
```

```
####################
##combine data
####################
#concentration norm; average experiments
        if(length(RNAinput) > 1){

#pAnorm; average inputs; keep experiments separate
        pANormInputAveList<-lapply(pANormList, function(i){i[,1:4]})
        names(pANormInputAveList)<-names(pANormList)
        for(x in 1:length(pANormInputAveList)){
        for(i in 1:dim(pANormInputAveList[[x]])[1]){
                pANormInputAveList[[x]][i,3]<-mean(unlist(
                        pANormList[[x]][i,3:(3+length(RNAinput)-1)]))
                pANormInputAveList[[x]][i,4]<-sd(unlist(
                        pANormList[[x]][i,3:(3+length(RNAinput)-1)]))
                pANormInputAveList[[x]][i,5]<-
                        (pANormInputAveList[[x]][i,4]/
                        pANormInputAveList[[x]][i,3])}
                colnames(pANormInputAveList[[x]])[3:5]<-
                        c("pAnorm","error","PercentError")}

if(length(dataList) > 1){
#inputNorm; average inputs and experiments
        inputNormExpAve<-lapply(1:length(unique(identifier)), function(x){
                return(inputNormList[[1]])})
        for(x in 1:length(inputNormExpAve)){
        for(j in 3:(3+length(RNAinput)-1)){
        for(i in 1:dim(inputNormExpAve[[x]])[1]){
                inputNormExpAve[[x]][i,j]<-mean(sapply(which(
                        identifier == unique(identifier)[x]), function(k){
                                return(inputNormList[[k]][i,j])}))
                inputNormExpAve[[x]][i,(j+length(RNAinput))]<-sd(
                        sapply(which(identifier == unique(identifier)[x]), function(k){
                                return(inputNormList[[k]][i,j])}))
                inputNormExpAve[[x]][i,(j+2*length(RNAinput))]<-(
                                inputNormExpAve[[x]][i,(j+length(RNAinput))]/
                                inputNormExpAve[[x]][i,j])}
                names(inputNormExpAve[[x]])[(j+2*length(RNAinput))]<-
                                paste(RNAinput[j-2], "percentError",sep="")}}

#pAnorm; average inputs and experiments
        pANormInputExpAve<-lapply(1:length(unique(identifier)), function(x){
                return(pANormInputAveList[[1]])})
        for(x in 1:length(pANormInputExpAve)){
        for(i in 1:dim(pANormInputExpAve[[x]])[1]){
                pANormInputExpAve[[x]][i,3]<-mean(
                        sapply(which(identifier == unique(identifier)[x]), function(k){
                                return(pANormInputAveList[[k]][i,3])}))
                pANormInputExpAve[[x]][i,4]<-sd(
                        sapply(which(identifier == unique(identifier)[x]), function(k){
                                return(pANormInputAveList[[k]][i,3])}))
                pANormInputExpAve[[x]][i,5]<-
                        (pANormInputExpAve[[x]][i,4]/
                        pANormInputExpAve[[x]][i,3])}
                names(pANormInputExpAve[[x]])[5]<-"percentError"
                write.table(pANormInputExpAve,
                        file = paste(dir,"plots/rData/", unique(identifier)[x],
                        paste(filenames, collapse="_"),"pANormInputExpAveData.tab",
                        sep=""), sep = "\t")}}}
```

```
#pAnorm; average experiments, but keep inputs separately
#use identifier; to know which ones to average
        if(length(dataList) > 1){
        pANormExpAve<-lapply(1:length(unique(identifier)), function(x){
                return(pANormList[[1]])})
        for(x in 1:length(pANormExpAve)){
        for(j in 3:(3+length(RNAinput)-1)){
        for(i in 1:dim(pANormExpAve[[x]])[1]){
                pANormExpAve[[x]][i,j]<-mean(
                        sapply(which(identifier == unique(identifier)[x]), function(k){
                                return(pANormList[[k]][i,j])}))
                pANormExpAve[[x]][i,(j+length(RNAinput))]<-sd(
                        sapply(which(identifier == unique(identifier)[x]), function(k){
                                return(pANormList[[k]][i,j])}))
                pANormExpAve[[x]][i,(j+2*length(RNAinput))]<-
                        (pANormExpAve[[x]][i,(j+length(RNAinput))]/
                        pANormExpAve[[x]][i,j])}
                names(pANormExpAve[[x]])[(j+2*length(RNAinput))]<-
                        paste(RNAinput[j-2], "percentError",sep="")}}
        names(pANormExpAve)<-unique(identifier)}

####################
##fit linear models
####################
#TNL~[RNA]; experiments separate
        if(length(RNAinput) > 1){
        #raw; list with a model for each pA length
        inputLmList<-lapply(dataList, function(x){
                lapply(1:dim(dataList[[1]])[1], function(i){
                        lm(unlist(x[i,3:(3+length(RNAinput)-1)]) ~ RNAinput)})})
        #normalized; list with a model for each pA length
        inputNormLmList<-lapply(inputNormList, function(x){
                lapply(1:dim(inputNormList[[1]])[1], function(i){
                        lm(unlist(x[i,3:(3+length(RNAinput)-1)]) ~ RNAinputNorm)})})

#TNL~[RNA]; experiments combined
        #normalized; list with a model for each pA length
        if(length(dataList) > 1){
        inputNormExpAveLm<-lapply(1:length(inputNormExpAve), function(x){
                lapply(1:dim(inputNormExpAve[[x]])[1], function(i){
                        lm(unlist(inputNormExpAve[[x]][i,3:(3+length(RNAinput)-1)]) ~
                                RNAinputNorm)})})}}

#TNL~pALength; inputs and experiments separate
        #raw; list with a model for each input
        pALmList<-lapply(dataList, function(x){
                lapply(3:(3+length(RNAinput)-1), function(i){
                        lm(unlist(x[,i]) ~ pATailLength)})})
        #norm; list with a model for each input
        pANormLmList<-lapply(pANormList, function(x){
                lapply(3:(3+length(RNAinput)-1), function(i){
                        lm(unlist(x[,i]) ~ pALengthNorm)})})
        #pA > 10; list with a model for each input for tails greater than 10NT
        pANormTenUpLmList<-lapply(pANormList, function(x){
                lapply(3:(3+length(RNAinput)-1), function(i){
                        lm(unlist(x[which(pALengthNorm > 0.1),i]) ~
                                pALengthNorm[which(pALengthNorm > 0.1)])})})

#TNL~pALength; combine inputs but not experiments
```

```
        if(length(RNAinput) > 1){
        #list with a model for each experiment
        pANormInputAveLmList<-lapply(pANormInputAveList, function(x){
                lm(unlist(x[,3]) ~ pALengthNorm)})
        #pA > 10; list with a model for each experiment for tails greater than 10NT
        pANormTenUpInputAveLmList<-lapply(pANormInputAveList, function(x){
                lm(unlist(x[which(pALengthNorm > 0.1),3]) ~
                        pALengthNorm[which(pALengthNorm > 0.1)])})}

#TNL~pALength; combine experiments, but not inputs
        #list with models for each input
        if(length(dataList) > 1){
        pANormExpAveLmList<-lapply(1:length(pANormExpAve), function(x){
                lapply(3:(3+length(RNAinput)-1), function(i){
                        lm(unlist(pANormExpAve[[x]][,i]) ~ pALengthNorm)})})
        #pA > 10; list with models for each input, for tails greater than 10NT
        pANormTenUpExpAveLmList<-lapply(1:length(pANormExpAve), function(x){
                lapply(3:(3+length(RNAinput)-1), function(i){
                        lm(unlist(pANormExpAve[[x]][which(pALengthNorm > 0.1),i]) ~
                                pALengthNorm[which(pALengthNorm > 0.1)])})})}

#TNL~pALength; experiments and inputs combined
        if(length(dataList) > 1 & length(RNAinput) > 1){
        pANormInputExpAveLm<-lapply(1:length(pANormInputExpAve), function(x){
                lm(pANormInputExpAve[[x]][,3] ~ pALengthNorm)})

        #pA > 10; model for tails greater than 10NT
        pANormTenUpInputExpAveLm<-lapply(1:length(pANormInputExpAve), function(x){
                lm(pANormInputExpAve[[x]][which(pALengthNorm > 0.1),3] ~
                        pALengthNorm[which(pALengthNorm > 0.1)])})}

####################
##make plots
####################
#color schemes and constants
#       par(las=3, mfrow = c(2,2))
#       pie(rep(1,12), col=rainbow(12), main = "Rainbow")
#       pie(rep(1,12), col=heat.colors(12), main = "heat.colors")
#       pie(rep(1,12), col=topo.colors(12), main = "topo.colors")
#       pie(rep(1,12), col=cm.colors(12), main = "cm.colors")

        pAcolors<-rev(c(terrain.colors(n=dim(dataList[[1]])[1]+1))[
                1:dim(dataList[[1]])[1]])
        inputColors<-c(topo.colors(n = length(RNAinput)))
        expColors<-c(rainbow(n = max(sapply(1:length(unique(identifier)), function(x){
                        length(which(identifier == unique(identifier)))}), na.rm = TRUE)))
        idenColors<-c("dodgerblue3", "red2")
#       idenColors<-c(cm.colors(n = length(unique(identifier)), alpha = 1))

        title = NA
        if(length(filenames)<2){title = filenames[1]}
        if(length(filenames)>1){title = paste(
                        sub("_.*", "", filenames[1]), paste(sapply(2:length(filenames),
                function(i){
        sub("^.*IVT", "", filenames[i])}), collapse = "_"), sep = "_")}
        if(subtitle == FALSE){
        if(length(filenames)<2){subtitle = sub("_.*", "",filenames[1])}
        if(length(filenames)>1){subtitle = paste(
                        sub("_.*", "", filenames[1]), paste(sapply(2:length(filenames),
                function(i){
```

```
                        sub("_.*", "", sub("^.*IVT", "", filenames[i]))}), collapse = "+"),
                sep = "+")}}
        pALabels = NA
        if(class(dataList[[1]][,2]) == "integer"){
                pALabels<-paste("pA(",dataList[[1]][,2], ")", sep = "")}
        if(class(dataList[[1]][,2]) == "character"){
                pALabels<-sub("^.*pA","pA",dataList[[1]][,2])}

##################
##################
        pdf(file = paste(dir,"plots/", title, "Plots.pdf", sep=""))
        par(las=3, mfrow = c(2,2))
        source(paste(dir,"code/makeBarPlot.R", sep=""))


##################
##################
#visualize errors
        #individual experiments; inputs separate
        sapply(1:length(dataList), function(y){
        makeBarPlot(
        data =
                cbind(
                sapply((2+2*length(RNAinput)+1):(2+3*length(RNAinput)), function(j){
                        dataList[[y]][,j]}),
                sapply((2+2*length(RNAinput)+1):(2+3*length(RNAinput)), function(j){
                        pANormList[[y]][,j]})),
                errors = NULL, ylim = c(0,0.5), cols = pAcolors,
                xlab = "",     ylab = "Percent Errors", main = "Percent Errors",
                subtitle = filenames[y],
                names = c(paste(RNAinput,"WithinExpDups",sep=""),
                paste(RNAinput,"WithinExpDups(pAnorm)",sep="")))
                legend(x = "topleft", legend = pALabels, col = pAcolors,
                lty = 1, lwd = 2, cex = 0.5)})

        #one plot for each identifier; combine experiments and inputs within;
        #plot max of both in each category
        sapply(1:length(unique(identifier)),function(x){
        par(las=3, mfrow = c(1,1))
        makeBarPlot(
                data =
                        cbind(
                        apply(sapply((2+2*length(RNAinput)+1):(2+3*length(RNAinput)), function(j){
                                apply(sapply(dataList[which(
                                        identifier == unique(identifier)[x])], function(y){
                                        return(y[,j])}), 1, max, na.rm = TRUE)}), 1, max,
na.rm=TRUE),
                        apply(sapply((2+2*length(RNAinput)+1):(2+3*length(RNAinput)), function(j){
                                apply(sapply(pANormList[which(
                                        identifier == unique(identifier)[x])], function(y){
                                        return(y[,j])}), 1, max, na.rm = TRUE)}),1, max,
na.rm=TRUE),
                        if(length(RNAinput) > 1){apply(sapply(pANormInputAveList[which(
                                identifier == unique(identifier)[x])], function(y){
                                        return(y[,5])}), 1, max, na.rm = TRUE)}else NA,
                        if(length(dataList) > 1 & length(RNAinput) > 1){
                                        apply(sapply(pANormInputExpAve[x], function(y){
                                                return(y[,5])}), 1, max, na.rm = TRUE)}else
                                if(length(dataList) > 1){
                                        apply(sapply(pANormExpAve[x], function(y){
```

```
                                                    return(y[,5])}), 1, max, na.rm = TRUE)} else
NA),
                errors = NULL, ylim = c(0,0.5), cols = pAcolors,
                xlab = "",     ylab = "Pecent Errors", main = "Percent Errors",
                subtitle = paste(unique(identifier)[x], " ", subtitle, sep = ""),
                names = c("WithinExpDups", "WithinExpDups(pAnorm)","BtwnInputs","BtwnExp"))
        legend(x = "topleft", legend = pALabels, col = pAcolors,
                lty = 1, lwd = 2, cex = 0.5)})
        par(las=3, mfrow = c(2,2))

##################
##################
#TNL~[RNA]
if(length(RNAinput) > 1){
        #XYscatter function
        xyRNAconcentration<-function(x, xLm, main, subtitle){
        plot(RNAinputNorm, unlist(x[1,3:(3+length(RNAinput)-1)]), pch = 1,
                col = pAcolors[1], ylab = "RLU (Fraction of Max Input)",
                xlab = "RNA Input (Fraction of Max)", ylim = c(0, 1.25), main = main)
        mtext(subtitle, side = 3, cex = 0.75, las = 0, padj = -1)
        errbar(RNAinputNorm, unlist(x[1,3:(3+length(RNAinput)-1)]),
                yplus = unlist(x[1,3:(3+length(RNAinput)-1)]) +
                        unlist(x[1,(2+length(RNAinput)+1):(2+2*length(RNAinput))]),
                yminus = unlist(x[1,3:(3+length(RNAinput)-1)]) -
                        unlist(x[1,(2+length(RNAinput)+1):(2+2*length(RNAinput))]),
                col = pAcolors[1], add = TRUE)
        abline(xLm[[1]], lwd = 2, col = pAcolors[1])
        sapply(2:dim(x)[1], function(i){
                points(RNAinputNorm, unlist(x[i,3:(3+length(RNAinput)-1)]),
                        pch = i, col = pAcolors[i])
                errbar(RNAinputNorm, unlist(x[i,3:(3+length(RNAinput)-1)]),
                        yplus = unlist(x[i,3:(3+length(RNAinput)-1)]) +
                                unlist(x[i,(2+length(RNAinput)+1):(2+2*length(RNAinput))]),
                        yminus = unlist(x[i,3:(3+length(RNAinput)-1)]) -
                                unlist(x[i,(2+length(RNAinput)+1):(2+2*length(RNAinput))]),
                        col = pAcolors[i], add = TRUE)
                abline(xLm[[i]], lwd = 2, col = pAcolors[i])})
        legend(x= "topleft", legend = sapply(1:dim(x)[1], function(i){
                        paste(pALabels[i], "; Slope = ",
                        round(summary(xLm[[i]])$coefficients[2], digits = 2), sep = "")}),
                        col = pAcolors, pch = c(1:length(inputNormLmList)), cex = 0.5)}

        #barchart keeping inputs and experiments separate
        sapply(1:length(inputNormList), function(i){
                makeBarPlot(data = as.matrix(inputNormList[[i]][,(3+length(RNAinput)-1):3]),
                        errors = as.matrix(inputNormList[[i]][,(2+2*length(RNAinput)):
                                (2+length(RNAinput)+1)]),
                        ylim = c(0,1.25), cols = pAcolors,
                        xlab = "RNA Input", ylab = "RLU (Fraction of Max Input)",
                        names = paste(rev(RNAinput),"ng",sep=""),
                        main = "Translation ~ [RNA]",
                        subtitle = sub("_.*", "", filenames[i]))
        sapply(rev(RNAinputNorm), function(j){abline(h = j, lty = 2, lwd = 2,
                col = rgb(25,25,25,75,max=225))})
        legend(x="topright", legend = pALabels,
                        col = pAcolors, lty = 1, lwd = 2, cex = 0.5)
        #XY scatter
        xyRNAconcentration(x = inputNormList[[i]], xLm = inputNormLmList[[i]],
                                                    main = "Translation ~ [RNA]",
                                                    subtitle = filenames[i])})
```

```
        #barchart keeping inputs separate (but combining experiments)
        if(length(dataList)>1){
        sapply(1:length(inputNormExpAve), function(x){
        makeBarPlot(data = as.matrix(inputNormExpAve[[x]][,(3+length(RNAinput)-1):3]),
                errors = as.matrix(inputNormExpAve[[x]][,
                        (2+2*length(RNAinput)):(2+length(RNAinput)+1)]),
                ylim = c(0, 1.25), cols = pAcolors, xlab = "RNA Input",
                ylab = "RLU (Fraction of Max Input)",
                names = paste(rev(RNAinput), "ng", sep = ""),
                main = "Translation ~ [RNA]", subtitle = paste(
                        unique(identifier)[x], " ", subtitle, sep = ""))
        sapply(rev(RNAinputNorm), function(i){abline(h = i, lty = 2, lwd = 2,
                col = rgb(25,25,25,75,max=225))})
        legend(x="topright", legend = pALabels,
                        col = pAcolors, lty = 1, lwd = 2, cex = 0.5)
        #XY scatter of combined data
        xyRNAconcentration(x = inputNormExpAve[[x]], xLm = inputNormExpAveLm[[x]],
                                                        main = "Translation ~ [RNA]", subtitle =
subtitle)})}
        par(las=3, mfrow = c(2,2))}

##################
##################
##TNL~pA tail length
#XYscatter function
        xypALength<-function(data, dataIndecies, errorIndecies, pALengths,
                xLm, main, addLine, dataLabels, subtitle,
                pAcols, otherCols, yLim, ylab){
                #plot the first input amount
                plot(pALengths, unlist(data[,dataIndecies[1]]), pch = 16, cex = 1.5,
                        col = otherCols[1], ylab = ylab,
                        xlab = "pA Length (Fraction of Max)", ylim = yLim, main = main)
                mtext(subtitle, side = 3, cex = 0.75, las = 0, padj = -1)
                errbar(pALengths, unlist(data[,dataIndecies[1]]),
                        yplus = unlist(data[,dataIndecies[1]]) +
                                unlist(data[,errorIndecies[1]]),
                        yminus = unlist(data[,dataIndecies[1]]) -
                                unlist(data[,errorIndecies[1]]),
                        col = rgb(25,25,25,75,max=255), add = TRUE)
                #add linear fit for first input if desired
                #(allows to use same code to make plots +/- lines)
                if(addLine==TRUE){
                        if(class(xLm) == "list"){
                        abline(xLm[[1]], lty = 2, lwd = 2,
                        col = rgb(t(col2rgb(otherCols[1])),alpha = 75, max = 255))}
                        if(class(xLm) == "lm"){
                        abline(xLm, lty = 2, lwd = 2,
                        col = rgb(t(col2rgb(otherCols[1])),alpha = 75, max = 255))}}
                #plot additional amounts
                if(length(dataIndecies) > 1){
                sapply(2:length(dataIndecies), function(i){
                        points(pALengths, unlist(data[,dataIndecies[i]]), pch = 16,
                                cex = 1.5, col = otherCols[i])
                        errbar(pALengths, unlist(data[,dataIndecies[i]]),
                                yplus = unlist(data[,dataIndecies[i]]) +
                                        unlist(data[,errorIndecies[i]]),
                                yminus = unlist(data[,dataIndecies[i]]) -
                                        unlist(data[,errorIndecies[i]]),
                                col = rgb(25,25,25,75,max=255), add = TRUE)
```

```
                #add lines for additional input amounts
                if(addLine==TRUE){
                        abline(xLm[[i]], lty = 2, lwd = 2,
                                col = rgb(t(col2rgb(otherCols[i])),alpha = 75, max = 255))}})}
                #legend with Rsq and slope
                if(addLine==TRUE){
                        if(class(xLm) == "list"){
                        legend(x= "topleft", legend = sapply(1:length(dataIndecies), function(i){
                                paste(dataLabels[i],
                                "Slope = ", round(summary(xLm[[i]])$coefficients[2], digits = 2),
                                "; Rsq = ", round(summary(xLm[[i]])$adj.r.squared, digits = 3),
sep="")}),
                                pch = 16, col = otherCols, cex = 0.75)}
                        if(class(xLm) == "lm"){
                        legend(x= "topleft", legend = paste(dataLabels,"Slope = ",
                                round(summary(xLm)$coefficients[2], digits = 2),
                                "; Rsq = ", round(summary(xLm)$adj.r.squared, digits = 3), sep=""),
                                pch = 16, col = otherCols, cex = 0.75)}}else
                #legend w/o statistics
                legend(x= "topleft", legend = dataLabels, pch = 16, col = otherCols,
                        cex = 0.75)}

#plot calls
#TNL~pALength; raw
#barplot: plot for each experiment, plot for each separate input
#XYscatter: plot for each experiment, inputs as separate lines
        sapply(1:length(dataList), function(x){
        sapply((2+length(RNAinput)):3, function(i){
        makeBarPlot(data = as.matrix(dataList[[x]][,i]), errors = as.matrix(
                        dataList[[x]][,(i+length(RNAinput))]),
                ylim = ylimRaw, cols = pAcolors, ylab = "RLU (raw)", xlab = "",
                names = pALabels,main = "Translation ~ pA Length",
                subtitle = paste(filenames[x], "; ", RNAinput[i-2], "ng input", sep=""))})
        xypALength(data = dataList[[x]], dataIndecies = c(3:(3+length(RNAinput)-1)),
                errorIndecies = c((2+length(RNAinput)+1):(2+2*length(RNAinput))), yLim = ylimRaw,
                pALengths = pATailLength, xLm = pALmList[[x]],
                dataLabels = paste(RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length", subtitle = filenames[x],
                pAcols = pAcolors, otherCols = inputColors, addLine=TRUE, ylab = "RLU (raw)")
        par(las=3, mfrow = c(2,2))})

#TNL~pALength; norm
#barplot: plot for each experiment, plot for each separate input
#XYscatter: plot for each experiment, inputs as separate lines; for all data and just tails >
10NT
        sapply(1:length(dataList), function(x){
        sapply((2+length(RNAinput)):3, function(i){
        makeBarPlot(data = as.matrix(pANormList[[x]][,i]), errors = as.matrix(
                        pANormList[[x]][,(i+length(RNAinput))]),
                ylim = ylimNorm, cols = pAcolors, ylab = "RLU (raw)", xlab = "",
                names = pALabels, main = "Translation ~ pA Length", subtitle = paste(
                filenames[x], "; ", RNAinput[i-2], "ng input", sep=""))})
        xypALength(data = pANormList[[x]], dataIndecies = c(3:(3+length(RNAinput)-1)),
                errorIndecies = c((2+length(RNAinput)+1):(2+2*length(RNAinput))), yLim = ylimNorm,
                pALengths = pALengthNorm, xLm = pANormLmList[[x]], dataLabels = paste(
                        RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length", subtitle = filenames[x],
                pAcols = pAcolors, otherCols = inputColors, addLine=TRUE,
                ylab = "RLU (fraction of pA(100))")
        xypALength(data = pANormList[[x]][which(pALengthNorm > 0.1),],
```

```
                dataIndecies = c(3:(3+length(RNAinput)-1)),
                errorIndecies = c((2+length(RNAinput)+1):(2+2*length(RNAinput))),
                yLim = ylimNorm, pALengths = pALengthNorm[which(pALengthNorm > 0.1)],
                xLm = pANormTenUpLmList[[x]], dataLabels = paste(RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length; pA > 10NT",
                subtitle = filenames[x], pAcols = pAcolors, otherCols = inputColors, addLine=TRUE,
                ylab = "RLU (fraction of pA(100))")
        par(las=3, mfrow = c(2,2))})

#TNL~pALength; combine experiments but keep inputs separate
#barplot: one plot for each input
#XYscatter: one plot, inputs as separate lines; for all data and just tails > 10NT
        if(length(RNAinput) > 1 & length(dataList) > 1){
        sapply(1:length(pANormExpAve), function(x){
        sapply((2+length(RNAinput)):3, function(i){
        makeBarPlot(data = as.matrix(pANormExpAve[[x]][,i]), errors = as.matrix(
                        pANormExpAve[[x]][,(i+length(RNAinput))]),
                ylim = ylimNorm, cols = pAcolors, ylab = "RLU (fraction of pA(100))",
                xlab = "", names = pALabels, main = "Translation ~ pA Length",
                subtitle = paste(unique(identifier)[x], " ", subtitle, "; ",
                        RNAinput[i-2], "ng input", sep=""))})
        xypALength(data = pANormExpAve[[x]], dataIndecies = c(3:(3+length(RNAinput)-1)),
                errorIndecies = c((2+length(RNAinput)+1):(2+2*length(RNAinput))),
                yLim = ylimNorm, pALengths = pALengthNorm, xLm = pANormExpAveLmList[[x]],
                dataLabels = paste(RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length", subtitle = paste(unique(identifier)[x],
                        subtitle, sep = " "),
                pAcols = pAcolors, otherCols = inputColors, addLine=TRUE,
                        ylab = "RLU (fraction of pA(100))")
        xypALength(data = pANormExpAve[[x]][which(pALengthNorm > 0.1),],
                dataIndecies = c(3:(3+length(RNAinput)-1)),
                errorIndecies = c((2+length(RNAinput)+1):(2+2*length(RNAinput))),
                yLim = ylimNorm,pALengths = pALengthNorm[which(pALengthNorm > 0.1)],
                xLm = pANormTenUpExpAveLmList[[x]],
                dataLabels = paste(RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length; pA > 10NT",
                subtitle = paste(unique(identifier)[x], subtitle, sep = " "),
                pAcols = pAcolors, otherCols = inputColors,
                addLine=TRUE, ylab = "RLU (fraction of pA(100))")
        par(las=3, mfrow = c(2,2))})}

#TNL~pALength; combine inputs but keep experiments separate
#barchart: if more than one input, one plot for each experiment, inputs combined within
#XYscatter: single plot for each identifier, experiments as separate lines;
        if(length(RNAinput) < 2){
        sapply(1:length(unique(identifier)), function(x){
        par(las=3, mfrow = c(1,1))
        xypALength(
                data = cbind(
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                return(pANormList[[x]][,3])}),
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                return(pANormList[[x]][,4])})),
                dataIndecies = c(1:length(which(identifier == unique(identifier)[x]))),
                errorIndecies = c((length(which(identifier == unique(identifier)[x]))+1):
                                                        (2*length(which(identifier ==
unique(identifier)[x])))),
                pALengths = pALengthNorm, xLm = unlist(pANormLmList, recursive = FALSE),
                dataLabels = filenames[which(identifier == unique(identifier)[x])],
                yLim = ylimNorm, main = "Translation ~ pA Length",
```

```
                subtitle = paste(unique(identifier)[x], " ", RNAinput, "ng Input", sep = ""),
                pAcols = pAcolors, otherCols = expColors, addLine = TRUE,
                ylab = "RLU(fraction of pA(100))")
        #for pA length > 10
        xypALength(
                data = cbind(
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                return(pANormList[[x]][which(pALengthNorm > 0.1),3])}),
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                return(pANormList[[x]][which(pALengthNorm > 0.1),4])})),
                dataIndecies = c(1:length(which(identifier == unique(identifier)[x]))),
                errorIndecies = c((length(which(identifier == unique(identifier)[x]))+1):
                                                (2*length(which(identifier ==
unique(identifier)[x])))),
                pALengths = pALengthNorm[which(pALengthNorm > 0.1)],
                        xLm = unlist(pANormTenUpLmList, recursive = FALSE),
                dataLabels = filenames[which(identifier == unique(identifier)[x])],
                        yLim = ylimNorm, main = "Translation ~ pA Length; pA > 10NT",
                subtitle = paste(unique(identifier)[x], " ", RNAinput, "ng Input", sep = ""),
                pAcols = pAcolors, otherCols = expColors, addLine = TRUE,
                ylab = "RLU(fraction of pA(100))")})
        par(las=3, mfrow = c(2,2))}

        if(length(RNAinput) > 1){
        par(las=3, mfrow = c(2,2))
        sapply(1:length(dataList), function(x){
        makeBarPlot(data = as.matrix(pANormInputAveList[[x]][,3]),
                errors = as.matrix(pANormInputAveList[[x]][,4]),
                ylim = ylimNorm, cols = pAcolors, ylab = "RLU (fraction of pA(100))",
                xlab = "", names = pALabels, main = "Translation ~ pA Length",
                subtitle = paste(filenames[x], "; Inputs Combined", sep=""))})
        par(las=3, mfrow = c(1,1))
        sapply(1:length(unique(identifier)), function(x){
        xypALength(
                data = cbind(
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                return(pANormInputAveList[[x]][,3])}),
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                return(pANormInputAveList[[x]][,4])})),
                dataIndecies = c(1:length(which(identifier == unique(identifier)[x]))),
                errorIndecies = c((length(which(identifier == unique(identifier)[x]))+1):
                                                (2*length(which(identifier ==
unique(identifier)[x])))),
                pALengths = pALengthNorm, xLm = pANormInputAveLmList,
                dataLabels = filenames, yLim = ylimNorm,
                main = "Translation ~ pA Length",
                subtitle = paste(unique(identifier)[x], " ", subtitle,"; Inputs Combined",sep =
""),
                pAcols = pAcolors, otherCols = expColors, addLine = TRUE, ylab = "RLU(fraction of
pA(100))")
        xypALength(
                data = cbind(
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                        return(pANormInputAveList[[x]][which(pALengthNorm >
0.1),3])}),
                        sapply(which(identifier == unique(identifier)[x]), function(x){
                                        return(pANormInputAveList[[x]][which(pALengthNorm >
0.1),4])})),
                dataIndecies = c(1:length(which(identifier == unique(identifier)[x]))),
                errorIndecies = c((length(which(identifier == unique(identifier)[x]))+1):
```

```
                                                (2*length(which(identifier ==
unique(identifier)[x]))),
                pALengths = pALengthNorm[which(pALengthNorm > 0.1)],
                xLm = pANormTenUpInputAveLmList,
                dataLabels = filenames, yLim = ylimNorm,
                main = "Translation ~ pA Length; pA > 10NT",
                subtitle = paste(unique(identifier)[x], " ",
                        subtitle,"; Inputs Combined",sep = ""),
                pAcols = pAcolors, otherCols = expColors, addLine = TRUE,
                ylab = "RLU(fraction of pA(100))")})
        par(las=3, mfrow = c(2,2))}


#TNL~pALength; combine inputs and experiments
#barchart: single plot for each identifier
#XYscatter: single plot for each identifier
#XYscatter: single plot with identifiers as lines
        #multiple experiments single input
        if(length(dataList) > 1 & length(RNAinput) < 2){
        sapply(1:length(unique(identifier)), function(x){
        par(las=3, mfrow = c(1,1))
        makeBarPlot(data = as.matrix(pANormExpAve[[x]][,3]),
                errors = as.matrix(pANormExpAve[[x]][,4]),
                ylim = ylimNorm, cols = pAcolors, ylab = "RLU (fraction of pA(100))",
                xlab = "", names = pALabels, main = "Translation ~ pA Length",
                subtitle = paste(unique(identifier)[x], " ", subtitle,"; " ,RNAinput,
                        "ng; Exp Combined",sep = ""))
        xypALength(data = pANormExpAve[[x]], dataIndecies = c(3),
                errorIndecies = c(4), yLim = ylimNorm,
                pALengths = pALengthNorm, xLm = pANormExpAveLmList[[x]],
                dataLabels = paste(RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length", subtitle = paste(unique(identifier)[x], " ",
                subtitle,"; Exp Combined",sep = ""),
                pAcols = pAcolors, otherCols = "gray", addLine=TRUE,
                ylab = "RLU (fraction of pA(100))")
        xypALength(data = pANormExpAve[[x]][which(pALengthNorm > 0.1),],
                dataIndecies = c(3), errorIndecies = c(4), yLim = ylimNorm,
                pALengths = pALengthNorm[which(pALengthNorm > 0.1)],
                xLm = pANormTenUpExpAveLmList[[x]],
                dataLabels = paste(RNAinput, "ng Input", sep = ""),
                main = "Translation ~ pA Length; pA > 10NT",
                subtitle = paste(unique(identifier)[x], " ",
                        subtitle,"; Exp Combined",sep = ""),
                pAcols = pAcolors, otherCols = "gray", addLine=TRUE,
                ylab = "RLU (fraction of pA(100))")})
        xypALength(
                data = cbind(
                        cbind(sapply(pANormExpAve, function(x){return(x[,3])})),
                        cbind(sapply(pANormExpAve, function(x){return(x[,4])}))),
                dataIndecies = c(1:length(unique(identifier))),
                errorIndecies = c((length(unique(identifier))+1):
                                                (2*length(unique(identifier)))),
                pALengths = pALengthNorm, xLm = unlist(pANormExpAveLmList, recursive = FALSE),
                dataLabels = unique(identifier), yLim = ylimNorm,
                main = "Translation ~ pA Length",
                subtitle = paste(unique(identifier)[x], " ", RNAinput, "ng Input", sep = ""),
                pAcols = pAcolors, otherCols = idenColors, addLine = TRUE,
                        ylab = "RLU(fraction of pA(100))")}
        #multiple experiments; multiple inputs
        if(length(dataList) > 1 & length(RNAinput) > 1){
```

```
        sapply(1:length(unique(identifier)), function(x){
        par(las=3, mfrow = c(1,1))
        makeBarPlot(data = as.matrix(pANormInputExpAve[[x]][,3]),
                errors = as.matrix(pANormInputExpAve[[x]][,4]),
                ylim = ylimNorm, cols = pAcolors, ylab = "RLU (fraction of pA(100))",
                xlab = "", names = pALabels,
                main = "Translation ~ pA Length",
                subtitle = paste(unique(identifier)[x], " ",subtitle,
                        "; Inputs+Exp Combined",sep = ""))
        xypALength(data = pANormInputExpAve[[x]], dataIndecies = c(3),
                errorIndecies = c(4), yLim = ylimNorm,
                pALengths = pALengthNorm, xLm = pANormInputExpAveLm[[x]],
                dataLabels = subtitle, main = "Translation ~ pA Length",
                subtitle = paste(unique(identifier)[x], " ", subtitle,
                        "; Inputs+Exp Combined",sep = ""),
                pAcols = pAcolors, otherCols = "gray", addLine=TRUE,
                ylab = "RLU (fraction of pA(100))")
        xypALength(data = pANormInputExpAve[[x]][which(pALengthNorm > 0.1),],
                dataIndecies = c(3), errorIndecies = c(4),
                yLim = ylimNorm, pALengths = pALengthNorm[which(pALengthNorm > 0.1)],
                xLm = pANormTenUpInputExpAveLm[[x]],
                dataLabels = subtitle, main = "Translation ~ pA Length; pA > 10NT",
                subtitle = paste(unique(identifier)[x], " ", subtitle,
                "; Inputs+Exp Combined",sep = ""),
                pAcols = pAcolors, otherCols = "gray", addLine=TRUE,
                ylab = "RLU (fraction of pA(100))")})}


#TNL~pALength WT vs. d4G1; combine inputs and experiments
#barchart: WT/d4G1
#barchart: d4G1/WT
if(length(comparisons) > 1){
        sapply(seq(from = 1, to = length(comparisons), by =2), function(x){
        par(las=3, mfrow = c(1,1))
        c1<-comparisons[x]
        c2<-comparisons[x+1]
        makeBarPlot(
                data = as.matrix(pANormExpAve[[c1]][,3]/pANormExpAve[[c2]][,3]),
                errors = as.matrix((pANormExpAve[[c1]][,3]/pANormExpAve[[c2]][,3])*sqrt(
                                        (pANormExpAve[[c1]][,4]/pANormExpAve[[c1]][,3])^2+
                                        (pANormExpAve[[c2]][,4]/pANormExpAve[[c2]][,3])^2)),
                ylim = c(0,
                        max(as.matrix(pANormExpAve[[c1]][,3]/pANormExpAve[[c2]][,3])+
                                as.matrix((pANormExpAve[[c1]][,3]/pANormExpAve[[c2]][,3])*sqrt(
                                        (pANormExpAve[[c1]][,4]/pANormExpAve[[c1]][,3])^2+
                                        (pANormExpAve[[c2]][,4]/pANormExpAve[[c2]][,3])^2)), na.rm
= TRUE)+
                        0.1*max(as.matrix(pANormExpAve[[c1]][,3]/pANormExpAve[[c2]][,3])+
                                as.matrix((pANormExpAve[[c1]][,3]/pANormExpAve[[c2]][,3])*sqrt(
                                        (pANormExpAve[[c1]][,4]/pANormExpAve[[c1]][,3])^2+
                                        (pANormExpAve[[c2]][,4]/pANormExpAve[[c2]][,3])^2)), na.rm
= TRUE)),
                cols = pAcolors,
                ylab = paste(c1, " / ", c2, " (RLU (fraction of pA(100)))", sep = ""),
                xlab = "",
                names = pALabels,
                main = paste(c1, " vs. ", c2, " Translation ~ pA Length", sep = ""),
                subtitle = paste(subtitle, "Inputs+Exp Combined",sep = ""))})}
dev.off()}
```

BIBLIOGRAPHY

1. Mathews MB, Sonenberg N, Hershey JWB: **Origins and Principles of Translational Control**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 1-40.
2. Noller H: **Structure of the Bacterial Ribosome and Some Implications for Translational Regulation**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 41-58.
3. Taylor D, Frank J, Kinzy T: **Structure and Function of the Eukaryotic Ribosome and Elongation Factors**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 59-85.
4. Warner JR: **The economics of ribosome biosynthesis in yeast**. *Trends Biochem Sci* 1999, **24**(11):437-440.
5. Meisenberg G, Simmons WH: **Principles ofmedical biochemistry.** St.Louis, Missouri: Mosby; 1998.
6. Gavis E, Singer R, Hüttelmaier S: **Localized Translation through Messenger RNA Localization**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 689-717.
7. Thompson B, Wickens M, Kimble J: **Translational Control in Development**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 507-544.
8. Sonenberg N, Hinnebusch AG: **Regulation of translation initiation in eukaryotes: mechanisms and biological targets**. *Cell* 2009, **136**(4):731-745.
9. Morley S, Coldwell M: **Matters of Life and Death: Translation Initiation during Apoptosis**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 433-458.
10. Kozma S, Um S, Thomas G: **Translational Control in Metabolic Diseases: The Role of mTOR Signaling in Obesity and Diabetes**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 459-483.
11. Klann E, Richter J: **Translational Control of Synaptic Plasticity and Learning and Memory**. In: *Translational Control in Biology and Medicine.* Edited

by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 485-506.
12. Schneider R, Sonenberg N: **Translational Control in Cancer Development and Progression**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 401-431.
13. Ron D, Harding H: **eIF2α Phosphorylation in Cellular Stress Responses and Disease**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 345-368.
14. Mohr I, Peery T, Mathews M: **Protein Synthesis and Translational Control during Viral Infection**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 545-599.
15. Doudna JA, Sarnow P: **Translation Initiation by Viral Internal Ribosome Entry Sites**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 129-153.
16. Gygi SP, Rochon Y, Franza BR, Aebersold R: **Correlation between protein and mRNA abundance in yeast**. *Molecular and Cellular Biology* 1999, **19**(3):1720-1730.
17. Greenbaum D, Colangelo C, Williams K, Gerstein M: **Comparing protein abundance and mRNA expression levels on a genomic scale**. *Genome Biol* 2003, **4**(9):117.
18. Gry M, Rimini R, Strömberg S, Asplund A, Pontén F, Uhlén M, Nilsson P: **Correlations between RNA and protein expression profiles in 23 human cell lines**. *BMC genomics* 2009, **10**:365.
19. Ingolia NT, Ghaemmaghami S, Newman JRS, Weissman JS: **Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling**. *Science* 2009, **324**(5924):218-223.
20. de Godoy LMF, Olsen JV, Cox J, Nielsen ML, Hubner NC, Fröhlich F, Walther TC, Mann M: **Comprehensive mass-spectrometry-based proteome quantification of haploid versus diploid yeast**. *Nature* 2008, **455**(7217):1251-1254.
21. Pestova T, Lorsch J, Hellen C: **The Mechanism of Translation Initiation in Eukaryotes**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 87-128.
22. Hinnebusch A, Dever T, Asano K: **Mechanism of Translation Initiation in the Yeast Saccharomyces cerevisiae**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 225-268.

23. Ehrenberg M, Hauryliuk V, Crist C, Nakamura Y: **Translation Termination, the Prion [PSI+], and Ribosomal Recycling**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 173-196.
24. Tsung K, Inouye S, Inouye M: **Factors affecting the efficiency of protein synthesis in Escherichia coli. Production of a polypeptide of more than 6000 amino acid residues**. *J Biol Chem* 1989, **264**(8):4428-4433.
25. Fang P, Spevak CC, Wu C, Sachs MS: **A nascent polypeptide domain that can regulate translation elongation**. *Proc Natl Acad Sci USA* 2004, **101**(12):4059-4064.
26. Arava Y, Wang Y, Storey JD, Liu CL, Brown PO, Herschlag D: **Genome-wide analysis of mRNA translation profiles in Saccharomyces cerevisiae**. *Proc Natl Acad Sci USA* 2003, **100**(7):3889-3894.
27. Arava Y, Boas FE, Brown PO, Herschlag D: **Dissecting eukaryotic translation and its control by ribosome density mapping**. *Nucleic Acids Research* 2005, **33**(8):2421-2432.
28. Lodish HF, Jacobsen M: **Regulation of hemoglobin synthesis. Equal rates of translation and termination of - and -globin chains**. *J Biol Chem* 1972, **247**(11):3622-3629.
29. Walden WE, Godefroy-Colburn T, Thach RE: **The role of mRNA competition in regulating translation. I. Demonstration of competition in vivo**. *J Biol Chem* 1981, **256**(22):11739-11746.
30. Palmiter RD: **Quantitation of parameters that determine the rate of ovalbumin synthesis**. *Cell* 1975, **4**(3):189.
31. Shine J, Dalgarno L: **The 3'-terminal sequence of Escherichia coli 16S ribosomal RNA: complementarity to nonsense triplets and ribosome binding sites**. *Proc Natl Acad Sci USA* 1974, **71**(4):1342-1346.
32. Steitz JA, Jakes K: **How ribosomes select initiator regions in mRNA: base pair formation between the 3' terminus of 16S rRNA and the mRNA during initiation of protein synthesis in Escherichia coli**. *Proc Natl Acad Sci USA* 1975, **72**(12):4734-4738.
33. Laursen BS, Sørensen HP, Mortensen KK, Sperling-Petersen HU: **Initiation of protein synthesis in bacteria**. *Microbiol Mol Biol Rev* 2005, **69**(1):101-123.
34. Romby P, Springer M: **Translational Control in Prokaryotes**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 803-827.
35. Benelli D, Maone E, Londei P: **Two different mechanisms for ribosome/mRNA interaction in archaeal translation initiation**. *Mol Microbiol* 2003, **50**(2):635-643.
36. Benelli D, Londei P: **Begin at the beginning: evolution of translational initiation**. *Res Microbiol* 2009, **160**(7):493-501.

37. Kaberdina AC, Szaflarski W, Nierhaus KH, Moll I: **An unexpected type of ribosomes induced by kasugamycin: a look into ancestral times of protein synthesis?** *Mol Cell* 2009, **33**(2):227-236.
38. Hinnebusch AG: **eIF3: a versatile scaffold for translation initiation complexes**. *Trends Biochem Sci* 2006, **31**(10):553-562.
39. Kozak M: **Regulation of translation via mRNA structure in prokaryotes and eukaryotes**. *Gene* 2005, **361**:13-37.
40. Majumdar R, Bandyopadhyay A, Maitra U: **Mammalian translation initiation factor eIF1 functions with eIF1A and eIF3 in the formation of a stable 40 S preinitiation complex**. *J Biol Chem* 2003, **278**(8):6580-6587.
41. Kolupaeva VG, Unbehaun A, Lomakin IB, Hellen CUT, Pestova TV: **Binding of eukaryotic initiation factor 3 to ribosomal 40S subunits and its role in ribosomal dissociation and anti-association**. *RNA* 2005, **11**(4):470-486.
42. Asano K, Clayton J, Shalev A, Hinnebusch AG: **A multifactor complex of eukaryotic initiation factors, eIF1, eIF2, eIF3, eIF5, and initiator tRNA(Met) is an important translation initiation intermediate in vivo**. *Genes Dev* 2000, **14**(19):2534-2546.
43. Lomakin IB, Shirokikh NE, Yusupov MM, Hellen CUT, Pestova TV: **The fidelity of translation initiation: reciprocal activities of eIF1, IF3 and YciH**. *EMBO J* 2006, **25**(1):196-210.
44. Pestova TV, Borukhov SI, Hellen CU: **Eukaryotic ribosomes require initiation factors 1 and 1A to locate initiation codons**. *Nature* 1998, **394**(6696):854-859.
45. Pestova TV, Kolupaeva VG: **The roles of individual eukaryotic translation initiation factors in ribosomal scanning and initiation codon selection**. *Genes Dev* 2002, **16**(22):2906-2922.
46. Passmore LA, Schmeing TM, Maag D, Applefield DJ, Acker MG, Algire MA, Lorsch JR, Ramakrishnan V: **The eukaryotic translation initiation factors eIF1 and eIF1A induce an open conformation of the 40S ribosome**. *Mol Cell* 2007, **26**(1):41-50.
47. Lomakin IB, Kolupaeva VG, Marintchev A, Wagner G, Pestova TV: **Position of eukaryotic initiation factor eIF1 on the 40S ribosomal subunit determined by directed hydroxyl radical probing**. *Genes Dev* 2003, **17**(22):2786-2797.
48. Algire MA, Maag D, Lorsch JR: **Pi release from eIF2, not GTP hydrolysis, is the step controlled by start-site selection during eukaryotic translation initiation**. *Mol Cell* 2005, **20**(2):251-262.
49. Maag D, Fekete CA, Gryczynski Z, Lorsch JR: **A conformational change in the eukaryotic translation preinitiation complex and release of eIF1 signal recognition of the start codon**. *Mol Cell* 2005, **17**(2):265-275.

50. Fekete CA, Mitchell SF, Cherkasova VA, Applefield D, Algire MA, Maag D, Saini AK, Lorsch JR, Hinnebusch AG: **N- and C-terminal residues of eIF1A have opposing effects on the fidelity of start codon selection**. *EMBO J* 2007, **26**(6):1602-1614.
51. Hinnebusch A: **Mechanism and regulation of initiator methionyl-tRNA binding to ribosomes.** In: *Translational control of gene expression.* Edited by Sonenberg N, Hershey JWB, Mathews MB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2000: 185–243.
52. Chaudhuri J, Chowdhury D, Maitra U: **Distinct functions of eukaryotic translation initiation factors eIF1A and eIF3 in the formation of the 40 S ribosomal preinitiation complex**. *J Biol Chem* 1999, **274**(25):17975-17980.
53. Dever T, Dar A, Sicheri F: **Mechanism of Translation Initiation in the Yeast Saccharomyces cerevisiae**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 319-344.
54. Siridechadilok B, Fraser CS, Hall RJ, Doudna JA, Nogales E: **Structural roles for human translation factor eIF3 in initiation of protein synthesis**. *Science* 2005, **310**(5753):1513-1515.
55. Valásek L, Mathew AA, Shin B-S, Nielsen KH, Szamecz B, Hinnebusch AG: **The yeast eIF3 subunits TIF32/a, NIP1/c, and eIF5 make critical connections with the 40S ribosome in vivo**. *Genes Dev* 2003, **17**(6):786-799.
56. Pisarev AV, Kolupaeva VG, Yusupov MM, Hellen CUT, Pestova TV: **Ribosomal position and contacts of mRNA in eukaryotic translation initiation complexes**. *EMBO J* 2008, **27**(11):1609-1621.
57. Lamphear BJ, Kirchweger R, Skern T, Rhoads RE: **Mapping of functional domains in eukaryotic protein synthesis initiation factor 4G (eIF4G) with picornaviral proteases. Implications for cap-dependent and cap-independent translational initiation**. *J Biol Chem* 1995, **270**(37):21975-21983.
58. Imataka H, Sonenberg N: **Human eukaryotic translation initiation factor 4G (eIF4G) possesses two separate and independent binding sites for eIF4A**. *Molecular and Cellular Biology* 1997, **17**(12):6940-6947.
59. Asano K, Shalev A, Phan L, Nielsen K, Clayton J, Valásek L, Donahue TF, Hinnebusch AG: **Multiple roles for the C-terminal domain of eIF5 in translation initiation complex assembly and GTPase activation**. *EMBO J* 2001, **20**(9):2326-2337.
60. Yamamoto Y, Singh CR, Marintchev A, Hall NS, Hannig EM, Wagner G, Asano K: **The eukaryotic initiation factor (eIF) 5 HEAT domain mediates multifactor assembly and scanning with distinct interfaces to**

eIF1, eIF2, eIF3, and eIF4G. *Proc Natl Acad Sci USA* 2005, **102**(45):16164-16169.
61. Schütz P, Bumann M, Oberholzer AE, Bieniossek C, Trachsel H, Altmann M, Baumann U: **Crystal structure of the yeast eIF4A-eIF4G complex: an RNA-helicase controlled by protein-protein interactions**. *Proc Natl Acad Sci USA* 2008, **105**(28):9564-9569.
62. Gingras AC, Raught B, Sonenberg N: **eIF4 initiation factors: effectors of mRNA recruitment to ribosomes and regulators of translation**. *Annu Rev Biochem* 1999, **68**:913-963.
63. Svitkin YV, Pause A, Haghighat A, Pyronnet S, Witherell G, Belsham GJ, Sonenberg N: **The requirement for eukaryotic initiation factor 4A (eIF4A) in translation is in direct proportion to the degree of mRNA 5' secondary structure**. *RNA* 2001, **7**(3):382-394.
64. Marintchev A, Edmonds KA, Marintcheva B, Hendrickson E, Oberer M, Suzuki C, Herdy B, Sonenberg N, Wagner G: **Topology and regulation of the human eIF4A/4G/4H helicase complex in translation initiation**. *Cell* 2009, **136**(3):447-460.
65. Abramson RD, Dever TE, Merrick WC: **Biochemical evidence supporting a mechanism for cap-independent and internal initiation of eukaryotic mRNA**. *J Biol Chem* 1988, **263**(13):6016-6019.
66. Rogers GW, Richter NJ, Lima WF, Merrick WC: **Modulation of the helicase activity of eIF4A by eIF4B, eIF4H, and eIF4F**. *J Biol Chem* 2001, **276**(33):30914-30922.
67. Dmitriev SE, Terenin IM, Dunaevsky YE, Merrick WC, Shatsky IN: **Assembly of 48S translation initiation complexes from purified components with mRNAs that have some base pairing within their 5' untranslated regions**. *Molecular and Cellular Biology* 2003, **23**(24):8925-8933.
68. Altmann M, Müller PP, Wittmer B, Ruchti F, Lanker S, Trachsel H: **A Saccharomyces cerevisiae homologue of mammalian translation initiation factor 4B contributes to RNA helicase activity**. *EMBO J* 1993, **12**(10):3997-4003.
69. Coppolecchia R, Buser P, Stotz A, Linder P: **A new yeast translation initiation factor suppresses a mutation in the eIF-4A RNA helicase**. *EMBO J* 1993, **12**(10):4005-4011.
70. Vornlocher HP, Hanachi P, Ribeiro S, Hershey JW: **A 110-kilodalton subunit of translation initiation factor eIF3 and an associated 135-kilodalton protein are encoded by the Saccharomyces cerevisiae TIF32 and TIF31 genes**. *J Biol Chem* 1999, **274**(24):16802-16812.
71. Méthot N, Song MS, Sonenberg N: **A region rich in aspartic acid, arginine, tyrosine, and glycine (DRYG) mediates eukaryotic initiation factor 4B (eIF4B) self-association and interaction with eIF3**. *Molecular and Cellular Biology* 1996, **16**(10):5328-5334.

72. Sonenberg N, Morgan MA, Merrick WC, Shatkin AJ: **A polypeptide in eukaryotic initiation factors that crosslinks specifically to the 5'-terminal cap in mRNA**. *Proc Natl Acad Sci USA* 1978, **75**(10):4843-4847.
73. Sonenberg N, Rupprecht KM, Hecht SM, Shatkin AJ: **Eukaryotic mRNA cap binding protein: purification by affinity chromatography on sepharose-coupled m7GDP**. *Proc Natl Acad Sci USA* 1979, **76**(9):4345-4349.
74. Altmann M, Edery I, Sonenberg N, Trachsel H: **Purification and characterization of protein synthesis initiation factor eIF-4E from the yeast Saccharomyces cerevisiae**. *Biochemistry* 1985, **24**(22):6085-6089.
75. Mader S, Lee H, Pause A, Sonenberg N: **The translation initiation factor eIF-4E binds to a common motif shared by the translation factor eIF-4 gamma and the translational repressors 4E-binding proteins**. *Molecular and Cellular Biology* 1995, **15**(9):4990-4997.
76. Haghighat A, Sonenberg N: **eIF4G dramatically enhances the binding of eIF4E to the mRNA 5'-cap structure**. *J Biol Chem* 1997, **272**(35):21677-21680.
77. Ptushkina M, von der Haar T, Vasilescu S, Frank R, Birkenhäger R, McCarthy JE: **Cooperative modulation by eIF4G of eIF4E-binding to the mRNA 5' cap in yeast involves a site partially shared by p20**. *EMBO J* 1998, **17**(16):4798-4808.
78. von Der Haar T, Ball PD, McCarthy JE: **Stabilization of eukaryotic initiation factor 4E binding to the mRNA 5'-Cap by domains of eIF4G**. *J Biol Chem* 2000, **275**(39):30551-30555.
79. Gross JD, Moerke NJ, von der Haar T, Lugovskoy AA, Sachs AB, McCarthy JE, Wagner G: **Ribosome loading onto the mRNA cap is driven by conformational coupling between eIF4G and eIF4E**. *Cell* 2003, **115**(6):739-750.
80. Wells SE, Hillner PE, Vale RD, Sachs AB: **Circularization of mRNA by eukaryotic translation initiation factors**. *Mol Cell* 1998, **2**(1):135-140.
81. Gallie DR: **The cap and poly(A) tail function synergistically to regulate mRNA translational efficiency**. *Genes Dev* 1991, **5**(11):2108-2116.
82. Tarun SZ, Sachs AB: **A common function for mRNA 5' and 3' ends in translation initiation in yeast**. *Genes Dev* 1995, **9**(23):2997-3007.
83. Le H, Tanguay RL, Balasta ML, Wei CC, Browning KS, Metz AM, Goss DJ, Gallie DR: **Translation initiation factors eIF-iso4G and eIF-4B interact with the poly(A)-binding protein and increase its RNA binding activity**. *J Biol Chem* 1997, **272**(26):16247-16255.
84. Borman AM, Michel YM, Kean KM: **Biochemical characterisation of cap-poly(A) synergy in rabbit reticulocyte lysates: the eIF4G-PABP interaction increases the functional affinity of eIF4E for the capped mRNA 5'-end**. *Nucleic Acids Research* 2000, **28**(21):4068-4075.

85. Park E-H, Walker SE, Lee JM, Rothenburg S, Lorsch JR, Hinnebusch AG: **Multiple elements in the eIF4G1 N-terminus promote assembly of eIF4G1•PABP mRNPs in vivo**. *EMBO J* 2010.
86. Rogers GW, Richter NJ, Merrick WC: **Biochemical and kinetic characterization of the RNA helicase activity of eukaryotic initiation factor 4A**. *J Biol Chem* 1999, **274**(18):12236-12244.
87. Oberer M, Marintchev A, Wagner G: **Structural basis for the enhancement of eIF4A helicase activity by eIF4G**. *Genes Dev* 2005, **19**(18):2212-2223.
88. Korneeva NL, First EA, Benoit CA, Rhoads RE: **Interaction between the NH2-terminal domain of eIF4A and the central domain of eIF4G modulates RNA-stimulated ATPase activity**. *J Biol Chem* 2005, **280**(3):1872-1881.
89. Pöyry TAA, Kaminski A, Jackson RJ: **What determines whether mammalian ribosomes resume scanning after translation of a short upstream open reading frame?** *Genes Dev* 2004, **18**(1):62-75.
90. Jivotovskaya AV, Valásek L, Hinnebusch AG, Nielsen KH: **Eukaryotic translation initiation factor 3 (eIF3) and eIF2 can promote mRNA binding to 40S subunits independently of eIF4G in yeast**. *Molecular and Cellular Biology* 2006, **26**(4):1355-1372.
91. Das S, Maitra U: **Mutational analysis of mammalian translation initiation factor 5 (eIF5): role of interaction between the beta subunit of eIF2 and eIF5 in eIF5 function in vitro and in vivo**. *Molecular and Cellular Biology* 2000, **20**(11):3942-3950.
92. Maag D, Algire MA, Lorsch JR: **Communication between eukaryotic translation initiation factors 5 and 1A within the ribosomal pre-initiation complex plays a role in start site selection**. *Journal of Molecular Biology* 2006, **356**(3):724-737.
93. Choi SK, Lee JH, Zoll WL, Merrick WC, Dever TE: **Promotion of met-tRNAiMet binding to ribosomes by yIF2, a bacterial IF2 homolog in yeast**. *Science* 1998, **280**(5370):1757-1760.
94. Pestova TV, Lomakin IB, Lee JH, Choi SK, Dever TE, Hellen CU: **The joining of ribosomal subunits in eukaryotes requires eIF5B**. *Nature* 2000, **403**(6767):332-335.
95. Acker MG, Shin B-S, Dever TE, Lorsch JR: **Interaction between eukaryotic initiation factors 1A and 5B is required for efficient ribosomal subunit joining**. *J Biol Chem* 2006, **281**(13):8469-8475.
96. Shin B-S, Maag D, Roll-Mecak A, Arefin MS, Burley SK, Lorsch JR, Dever TE: **Uncoupling of initiation factor eIF5B/IF2 GTPase and translational activities by mutations that lower ribosome affinity**. *Cell* 2002, **111**(7):1015-1025.
97. Fringer JM, Acker MG, Fekete CA, Lorsch JR, Dever TE: **Coupled release of eukaryotic translation initiation factors 5B and 1A from 80S**

ribosomes following subunit joining. *Molecular and Cellular Biology* 2007, **27**(6):2384-2397.
98. Gebauer F, Hentze MW: **Molecular mechanisms of translational control**. *Nat Rev Mol Cell Biol* 2004, **5**(10):827-835.
99. Hentze M, Gebauer F, Preiss T: **cis-Regulatory Sequences and trans-Acting Factors in Translational Control**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 269-295.
100. Hinnebusch AG: **Translational regulation of GCN4 and the general amino acid control of yeast**. *Annu Rev Microbiol* 2005, **59**:407-450.
101. Raught B, Gingras A: **Signaling to Translation Initiation**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 369-400.
102. Churbanov A, Rogozin IB, Babenko VN, Ali H, Koonin EV: **Evolutionary conservation suggests a regulatory function of AUG triplets in 5'-UTRs of eukaryotic genes**. *Nucleic Acids Research* 2005, **33**(17):5512-5520.
103. Kozak M: **Comparison of initiation of protein synthesis in procaryotes, eucaryotes, and organelles**. *Microbiol Rev* 1983, **47**(1):1-45.
104. Gilbert WV, Zhou K, Butler TK, Doudna JA: **Cap-independent translation is required for starvation-induced differentiation in yeast**. *Science* 2007, **317**(5842):1224-1227.
105. Elroy-Stein O, Merrick W: **Translation Initiation Via Cellular Internal Ribosome Entry Sites**. In: *Translational Control in Biology and Medicine.* Edited by Mathews MB, Sonenberg N, Hershey JWB. Cold Spring Harbor, New York: Cold Spring Harbor Lab Press; 2007: 155-172.
106. Carthew RW, Sontheimer EJ: **Origins and Mechanisms of miRNAs and siRNAs**. *Cell* 2009, **136**(4):642-655.
107. Nomoto A, Lee YF, Wimmer E: **The 5' end of poliovirus mRNA is not capped with m7G(5')ppp(5')Np**. *Proc Natl Acad Sci USA* 1976, **73**(2):375-380.
108. Pelletier J, Sonenberg N: **Internal initiation of translation of eukaryotic mRNA directed by a sequence derived from poliovirus RNA**. *Nature* 1988, **334**(6180):320-325.
109. Chen CY, Sarnow P: **Initiation of protein synthesis by the eukaryotic translational apparatus on circular RNAs**. *Science* 1995, **268**(5209):415-417.
110. Pestova TV, Hellen CU, Shatsky IN: **Canonical eukaryotic initiation factors determine initiation of translation by internal ribosomal entry**. *Molecular and Cellular Biology* 1996, **16**(12):6859-6869.
111. Pestova TV, Shatsky IN, Hellen CU: **Functional dissection of eukaryotic initiation factor 4F: the 4A subunit and the central domain of the 4G subunit are sufficient to mediate internal entry of 43S**

preinitiation complexes. *Molecular and Cellular Biology* 1996, **16**(12):6870-6878.
112. Ghany MG, Strader DB, Thomas DL, Seeff LB, Diseases AAftSoL: **Diagnosis, management, and treatment of hepatitis C: an update**. *Hepatology* 2009, **49**(4):1335-1374.
113. Lauer GM, Walker BD: **Hepatitis C virus infection**. *N Engl J Med* 2001, **345**(1):41-52.
114. Deutsch M, Hadziyannis SJ: **Old and emerging therapies in chronic hepatitis C: an update**. *J Viral Hepat* 2008, **15**(1):2-11.
115. Rosenberg S: **Recent advances in the molecular biology of hepatitis C virus**. *Journal of Molecular Biology* 2001, **313**(3):451-464.
116. Friebe P, Bartenschlager R: **Genetic analysis of sequences in the 3' nontranslated region of hepatitis C virus that are important for RNA replication**. *Journal of Virology* 2002, **76**(11):5326-5338.
117. Song Y, Friebe P, Tzima E, Jünemann C, Bartenschlager R, Niepmann M: **The hepatitis C virus RNA 3'-untranslated region strongly enhances translation directed by the internal ribosome entry site**. *Journal of Virology* 2006, **80**(23):11579-11588.
118. Weinlich S, Hüttelmaier S, Schierhorn A, Behrens S-E, Ostareck-Lederer A, Ostareck DH: **IGF2BP1 enhances HCV IRES-mediated translation initiation via the 3'UTR**. *RNA* 2009, **15**(8):1528-1542.
119. Jopling CL, Schütz S, Sarnow P: **Position-dependent function for a tandem microRNA miR-122-binding site located in the hepatitis C virus RNA genome**. *Cell Host Microbe* 2008, **4**(1):77-85.
120. Jopling CL, Yi M, Lancaster AM, Lemon SM, Sarnow P: **Modulation of hepatitis C virus RNA abundance by a liver-specific MicroRNA**. *Science* 2005, **309**(5740):1577-1581.
121. Tsukiyama-Kohara K, Iizuka N, Kohara M, Nomoto A: **Internal ribosome entry site within hepatitis C virus RNA**. *Journal of Virology* 1992, **66**(3):1476-1483.
122. Wang C, Sarnow P, Siddiqui A: **Translation of human hepatitis C virus RNA in cultured cells is mediated by an internal ribosome-binding mechanism**. *Journal of Virology* 1993, **67**(6):3338-3344.
123. Brown EA, Zhang H, Ping LH, Lemon SM: **Secondary structure of the 5' nontranslated regions of hepatitis C virus and pestivirus genomic RNAs**. *Nucleic Acids Research* 1992, **20**(19):5041-5045.
124. Honda M, Brown EA, Lemon SM: **Stability of a stem-loop involving the initiator AUG controls the efficiency of internal initiation of translation on hepatitis C virus RNA**. *RNA* 1996, **2**(10):955-968.
125. Reynolds JE, Kaminski A, Carroll AR, Clarke BE, Rowlands DJ, Jackson RJ: **Internal initiation of translation of hepatitis C virus RNA: the ribosome entry site is at the authentic initiation codon**. *RNA* 1996, **2**(9):867-878.

126. Zhao WD, Wimmer E: **Genetic analysis of a poliovirus/hepatitis C virus chimera: new structure for domain II of the internal ribosomal entry site of hepatitis C virus**. *Journal of Virology* 2001, **75**(8):3719-3730.
127. Hellen CU, Pestova TV: **Translation of hepatitis C virus RNA**. *J Viral Hepat* 1999, **6**(2):79-87.
128. Rijnbrand RC, Lemon SM: **Internal ribosome entry site-mediated translation in hepatitis C virus replication**. *Curr Top Microbiol Immunol* 2000, **242**:85-116.
129. Fukushi S, Katayama K, Kurihara C, Ishiyama N, Hoshino FB, Ando T, Oya A: **Complete 5' noncoding region is necessary for the efficient internal initiation of hepatitis C virus RNA**. *Biochem Biophys Res Commun* 1994, **199**(2):425-432.
130. Rijnbrand R, Bredenbeek P, van der Straaten T, Whetter L, Inchauspé G, Lemon S, Spaan W: **Almost the entire 5' non-translated region of hepatitis C virus is required for cap-independent translation**. *FEBS Lett* 1995, **365**(2-3):115-119.
131. Honda M, Ping LH, Rijnbrand RC, Amphlett E, Clarke B, Rowlands D, Lemon SM: **Structural requirements for initiation of translation by internal ribosome entry within genome-length hepatitis C virus RNA**. *Virology* 1996, **222**(1):31-42.
132. Kamoshita N, Tsukiyama-Kohara K, Kohara M, Nomoto A: **Genetic analysis of internal ribosomal entry site on hepatitis C virus RNA: implication for involvement of the highly ordered structure and cell type-specific transacting factors**. *Virology* 1997, **233**(1):9-18.
133. Wang TH, Rijnbrand RC, Lemon SM: **Core protein-coding sequence, but not core protein, modulates the efficiency of cap-independent translation directed by the internal ribosome entry site of hepatitis C virus**. *Journal of Virology* 2000, **74**(23):11347-11358.
134. Kieft JS, Zhou K, Jubin R, Murray MG, Lau JY, Doudna JA: **The hepatitis C virus internal ribosome entry site adopts an ion-dependent tertiary fold**. *Journal of Molecular Biology* 1999, **292**(3):513-529.
135. Melcher SE, Wilson TJ, Lilley DMJ: **The dynamic nature of the four-way junction of the hepatitis C virus IRES**. *RNA* 2003, **9**(7):809-820.
136. Pestova TV, Shatsky IN, Fletcher SP, Jackson RJ, Hellen CU: **A prokaryotic-like mode of cytoplasmic eukaryotic ribosome binding to the initiation codon during internal translation initiation of hepatitis C and classical swine fever virus RNAs**. *Genes Dev* 1998, **12**(1):67-83.
137. Sizova DV, Kolupaeva VG, Pestova TV, Shatsky IN, Hellen CU: **Specific interaction of eukaryotic translation initiation factor 3 with the 5' nontranslated regions of hepatitis C virus and classical swine fever virus RNAs**. *Journal of Virology* 1998, **72**(6):4775-4782.

138. Kieft JS, Zhou K, Jubin R, Doudna JA: **Mechanism of ribosome recruitment by hepatitis C IRES RNA**. *RNA* 2001, **7**(2):194-206.
139. Reynolds JE, Kaminski A, Kettinen HJ, Grace K, Clarke BE, Carroll AR, Rowlands DJ, Jackson RJ: **Unique features of internal initiation of hepatitis C virus RNA translation**. *EMBO J* 1995, **14**(23):6010-6020.
140. Rijnbrand RC, Abbink TE, Haasnoot PC, Spaan WJ, Bredenbeek PJ: **The influence of AUG codons in the hepatitis C virus 5' nontranslated region on translation and mapping of the translation initiation window**. *Virology* 1996, **226**(1):47-56.
141. Kolupaeva VG, Pestova TV, Hellen CU: **An enzymatic footprinting analysis of the interaction of 40S ribosomal subunits with the internal ribosomal entry site of hepatitis C virus**. *Journal of Virology* 2000, **74**(14):6242-6250.
142. Lytle JR, Wu L, Robertson HD: **The ribosome binding site of hepatitis C virus mRNA**. *Journal of Virology* 2001, **75**(16):7629-7636.
143. Lytle JR, Wu L, Robertson HD: **Domains on the hepatitis C virus internal ribosome entry site for 40s subunit binding**. *RNA* 2002, **8**(8):1045-1055.
144. Spahn CM, Kieft JS, Grassucci RA, Penczek PA, Zhou K, Doudna JA, Frank J: **Hepatitis C virus IRES RNA-induced changes in the conformation of the 40s ribosomal subunit**. *Science* 2001, **291**(5510):1959-1962.
145. Otto GA, Puglisi JD: **The pathway of HCV IRES-mediated translation initiation**. *Cell* 2004, **119**(3):369-380.
146. Pestova TV, de Breyne S, Pisarev AV, Abaeva IS, Hellen CUT: **eIF2-dependent and eIF2-independent modes of initiation on the CSFV IRES: a common role of domain II**. *EMBO J* 2008, **27**(7):1060-1072.
147. Terenin IM, Dmitriev SE, Andreev DE, Shatsky IN: **Eukaryotic translation initiation machinery can operate in a bacterial-like mode without eIF2**. *Nat Struct Mol Biol* 2008, **15**(8):836-841.
148. Garaigorta U, Chisari FV: **Hepatitis C virus blocks interferon effector function by inducing protein kinase R phosphorylation**. *Cell Host Microbe* 2009, **6**(6):513-522.
149. Buratti E, Tisminetzky S, Zotti M, Baralle FE: **Functional analysis of the interaction between HCV 5'UTR and putative subunits of eukaryotic translation initiation factor eIF3**. *Nucleic Acids Research* 1998, **26**(13):3179-3187.
150. Ji H, Fraser CS, Yu Y, Leary J, Doudna JA: **Coordinated assembly of human translation initiation complexes by the hepatitis C virus internal ribosome entry site RNA**. *Proc Natl Acad Sci USA* 2004, **101**(49):16990-16995.

151. Locker N, Easton LE, Lukavsky PJ: **HCV and CSFV IRES domain II mediate eIF2 release during 80S ribosome assembly**. *EMBO J* 2007, **26**(3):795-805.
152. Berry KE, Waghray S, Doudna JA: **The HCV IRES pseudoknot positions the initiation codon on the 40S ribosomal subunit**. *RNA* 2010, **16**(8):1559-1569.
153. Hundsdoerfer P, Thoma C, Hentze MW: **Eukaryotic translation initiation factor 4GI and p97 promote cellular internal ribosome entry sequence-driven translation**. *Proc Natl Acad Sci USA* 2005, **102**(38):13421-13426.
154. Jackson RJ: **Alternative mechanisms of initiating translation of mammalian mRNAs**. *Biochem Soc Trans* 2005, **33**(Pt 6):1231-1241.
155. Gallie DR, Browning KS: **eIF4G functionally differs from eIFiso4G in promoting internal initiation, cap-independent translation, and translation of structured mRNAs**. *J Biol Chem* 2001, **276**(40):36951-36960.
156. Tarun SZ, Wells SE, Deardorff JA, Sachs AB: **Translation initiation factor eIF4G mediates in vitro poly(A) tail-dependent translation**. *Proc Natl Acad Sci USA* 1997, **94**(17):9046-9051.
157. Baker CC, Fuller MT: **Translational control of meiotic cell cycle progression and spermatid differentiation in male germ cells by a novel eIF4G homolog**. *Development* 2007, **134**(15):2863-2869.
158. Franklin-Dumont TM, Chatterjee C, Wasserman SA, Dinardo S: **A novel eIF4G homolog, Off-schedule, couples translational control to meiosis and differentiation in Drosophila spermatocytes**. *Development* 2007, **134**(15):2851-2861.
159. Contreras V, Richardson MA, Hao E, Keiper BD: **Depletion of the cap-associated isoform of translation factor eIF4G induces germline apoptosis in C. elegans**. *Cell Death Differ* 2008, **15**(8):1232-1242.
160. Bushell M, Sarnow P: **Hijacking the translation apparatus by RNA viruses**. *The Journal of Cell Biology* 2002, **158**(3):395-399.
161. Schneider RJ, Mohr I: **Translation initiation and viral tricks**. *Trends Biochem Sci* 2003, **28**(3):130-136.
162. Castelló A, Alvarez E, Carrasco L: **Differential cleavage of eIF4GI and eIF4GII in mammalian cells. Effects on translation**. *J Biol Chem* 2006, **281**(44):33206-33216.
163. Ramirez-Valle F, Braunstein S, Zavadil J, Formenti S, Schneider R: **eIF4GI links nutrient sensing by mTOR to cell proliferation and inhibition of autophagy**. *Journal of Cell Biology* 2008, **181**(2):293.
164. Lee SH, McCormick F: **p97/DAP5 is a ribosome-associated factor that facilitates protein synthesis and cell proliferation by modulating the synthesis of cell cycle proteins**. *EMBO J* 2006, **25**(17):4008-4019.

165. Brachmann CB, Davies A, Cost GJ, Caputo E, Li J, Hieter P, Boeke JD: **Designer deletion strains derived from Saccharomyces cerevisiae S288C: a useful set of strains and plasmids for PCR-mediated gene disruption and other applications**. *Yeast* 1998, **14**(2):115-132.
166. Searfoss AM, Wickner RB: **3' poly(A) is dispensable for translation**. *Proc Natl Acad Sci USA* 2000, **97**(16):9133-9137.
167. Guthrie C, Fink GR: **Guide to yeast genetics and molecular and cell biology**. Amsterdam ; Boston ; London: Academic Press; 2002.
168. Longtine MS, McKenzie A, Demarini DJ, Shah NG, Wach A, Brachat A, Philippsen P, Pringle JR: **Additional modules for versatile and economical PCR-based gene deletion and modification in Saccharomyces cerevisiae**. *Yeast* 1998, **14**(10):953-961.
169. Gietz RD, Woods RA: **Yeast transformation by the LiAc/SS Carrier DNA/PEG method**. *Methods Mol Biol* 2006, **313**:107-120.
170. !!! INVALID CITATION !!!
171. Mumberg D, Müller R, Funk M: **Yeast vectors for the controlled expression of heterologous proteins in different genetic backgrounds**. *Gene* 1995, **156**(1):119-122.
172. Browning KS, Webster C, Roberts JK, Ravel JM: **Identification of an isozyme form of protein synthesis initiation factor 4F in plants**. *J Biol Chem* 1992, **267**(14):10096-10100.
173. Goyer C, Altmann M, Lee HS, Blanc A, Deshmukh M, Woolford JL, Trachsel H, Sonenberg N: **TIF4631 and TIF4632: two yeast genes encoding the high-molecular-weight subunits of the cap-binding protein complex (eukaryotic initiation factor 4F) contain an RNA recognition motif-like sequence and carry out an essential function**. *Molecular and Cellular Biology* 1993, **13**(8):4860-4874.
174. Gradi A, Svitkin YV, Imataka H, Sonenberg N: **Proteolysis of human eukaryotic translation initiation factor eIF4GII, but not eIF4GI, coincides with the shutoff of host protein synthesis after poliovirus infection**. *Proc Natl Acad Sci USA* 1998, **95**(19):11089-11094.
175. Li L, Wang CC: **Capped mRNA with a single nucleotide leader is optimally translated in a primitive eukaryote, Giardia lamblia**. *J Biol Chem* 2004, **279**(15):14656-14664.
176. Li L, Wang CC: **Identification in the ancient protist Giardia lamblia of two eukaryotic translation initiation factor 4E homologues with distinctive functions**. *Eukaryotic Cell* 2005, **4**(5):948-959.
177. Tuteja R: **Identification and bioinformatics characterization of translation initiation complex eIF4F components and poly(A)-binding protein from Plasmodium falciparum**. *Commun Integr Biol* 2009, **2**(3):245-260.

178. Marintchev A, Wagner G: **eIF4G and CBP80 share a common origin and similar domain organization: implications for the structure and function of eIF4G**. *Biochemistry* 2005, **44**(37):12265-12272.
179. Hubbard TJ, Aken BL, Ayling S, Ballester B, Beal K, Bragin E, Brent S, Chen Y, Clapham P, Clarke L *et al*: **Ensembl 2009**. *Nucleic Acids Research* 2009, **37**(Database issue):D690-697.
180. Glanville JG, Kirshner D, Krishnamurthy N, Sjölander K: **Berkeley Phylogenomics Group web servers: resources for structural phylogenomic analysis**. *Nucleic Acids Research* 2007, **35**(Web Server issue):W27-32.
181. Letunic I, Bork P: **Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation**. *Bioinformatics* 2007, **23**(1):127-128.
182. Edgar RC: **MUSCLE: multiple sequence alignment with high accuracy and high throughput**. *Nucleic Acids Research* 2004, **32**(5):1792-1797.
183. Brengues M, Teixeira D, Parker R: **Movement of eukaryotic mRNAs between polysomes and cytoplasmic processing bodies**. *Science* 2005, **310**(5747):486-489.
184. Coyle SM, Gilbert WV, Doudna JA: **Direct link between RACK1 function and localization at the ribosome in vivo**. *Molecular and Cellular Biology* 2009, **29**(6):1626-1634.
185. Ritchie ME, Silver J, Oshlack A, Holmes M, Diyagama D, Holloway A, Smyth GK: **A comparison of background correction methods for two-colour microarrays**. *Bioinformatics* 2007, **23**(20):2700-2707.
186. Smyth GK, Speed T: **Normalization of cDNA microarray data**. *Methods* 2003, **31**(4):265-273.
187. Smyth GK: **Use of within-array replicate spots for assessing differential expression in microarray experiments**. *Bioinformatics* 2005, **21**(9):2067-2075.
188. Smyth GK: **Linear models and empirical bayes methods for assessing differential expression in microarray experiments**. *Statistical applications in genetics and molecular biology* 2004, **3**:Article3.
189. Gentleman R, Carey V, Huber W, Irizarry R, Dudoit S: **Bioinformatics and Computational Biology Solutions Using R and Bioconductor**: Springer; 2005.
190. Rasband WS: **ImageJ**. In. U. S. National Institutes of Health, Bethesda, Maryland, USA; 1997-2011.
191. Ghaemmaghami S, Huh W-K, Bower K, Howson RW, Belle A, Dephoure N, O'Shea EK, Weissman JS: **Global analysis of protein expression in yeast**. *Nature* 2003, **425**(6959):737-741.

192. Tarun SZ, Sachs AB: **Association of the yeast poly(A) tail binding protein with translation initiation factor eIF-4G**. *EMBO J* 1996, **15**(24):7168-7177.
193. Altmann M, Schmitz N, Berset C, Trachsel H: **A novel inhibitor of cap-dependent translation initiation in yeast: p20 competes with eIF4G for binding to eIF4E**. *EMBO J* 1997, **16**(5):1114-1121.
194. Dominguez D, Altmann M, Benz J, Baumann U, Trachsel H: **Interaction of translation initiation factor eIF4G with eIF4A in the yeast Saccharomyces cerevisiae**. *J Biol Chem* 1999, **274**(38):26720-26726.
195. Dominguez D, Kislig E, Altmann M, Trachsel H: **Structural and functional similarities between the central eukaryotic initiation factor (eIF)4A-binding domain of mammalian eIF4G and the eIF4A-binding domain of yeast eIF4G**. *Biochem J* 2001, **355**(Pt 1):223-230.
196. Lanker S, Müller PP, Altmann M, Goyer C, Sonenberg N, Trachsel H: **Interactions of the eIF-4F subunits in the yeast Saccharomyces cerevisiae**. *J Biol Chem* 1992, **267**(29):21167-21171.
197. Merrick WC, Hensold JO: **Analysis of Eukaryotic Translation in Purified and Semipurified Systems**. In: *Current Protocols in Cell Biology*. 2001.
198. Kuhn KM, DeRisi JL, Brown PO, Sarnow P: **Global and specific translational regulation in the genomic response of Saccharomyces cerevisiae to a rapid transfer from a fermentable to a nonfermentable carbon source**. *Molecular and Cellular Biology* 2001, **21**(3):916-927.
199. Sheth U, Parker R: **Decapping and decay of messenger RNA occur in cytoplasmic processing bodies**. *Science* 2003, **300**(5620):805-808.
200. von der Haar T, McCarthy JEG: **Intracellular translation initiation factor levels in Saccharomyces cerevisiae and their role in cap-complex function**. *Mol Microbiol* 2002, **46**(2):531-544.
201. Svitkin YV, Gradi A, Imataka H, Morino S, Sonenberg N: **Eukaryotic initiation factor 4GII (eIF4GII), but not eIF4GI, cleavage correlates with inhibition of host cell protein synthesis after human rhinovirus infection**. *Journal of Virology* 1999, **73**(4):3467-3472.
202. Bovee ML, Lamphear BJ, Rhoads RE, Lloyd RE: **Direct cleavage of eIF4G by poliovirus 2A protease is inefficient in vitro**. *Virology* 1998, **245**(2):241-249.
203. Xu Z, Wei W, Gagneur J, Perocchi F, Clauder-Münster S, Camblong J, Guffanti E, Stutz F, Huber W, Steinmetz LM: **Bidirectional promoters generate pervasive transcription in yeast**. *Nature* 2009, **457**(7232):1033-1037.
204. Coldwell MJ, Morley SJ: **Specific isoforms of translation initiation factor 4GI show differences in translational activity**. *Molecular and Cellular Biology* 2006, **26**(22):8448-8460.

205. Spellman P, Sherlock G, Zhang M, Iyer V: **Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by ...**. *Mol Biol Cell* 1998.
206. Berset C, Trachsel H, Altmann M: **The TOR (target of rapamycin) signal transduction pathway regulates the stability of translation initiation factor eIF4G in the yeast Saccharomyces cerevisiae**. *Proc Natl Acad Sci USA* 1998, **95**(8):4264-4269.
207. Clarkson BK, Gilbert WV, Doudna JA: **Functional overlap between eIF4G isoforms in Saccharomyces cerevisiae**. *PLoS ONE* 2010, **5**(2):e9114.
208. Jorgensen P, Rupes I, Sharom JR, Schneper L, Broach JR, Tyers M: **A dynamic transcriptional network communicates growth potential to ribosome synthesis and critical cell size**. *Genes Dev* 2004, **18**(20):2491-2505.
209. Boyle EI, Weng S, Gollub J, Jin H, Botstein D, Cherry JM, Sherlock G: **GO::TermFinder--open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes**. *Bioinformatics* 2004, **20**(18):3710-3715.
210. Powers T, Walter P: **Regulation of ribosome biogenesis by the rapamycin-sensitive TOR-signaling pathway in Saccharomyces cerevisiae**. *Mol Biol Cell* 1999, **10**(4):987-1000.
211. Wilmes GM, Bergkessel M, Bandyopadhyay S, Shales M, Braberg H, Cagney G, Collins SR, Whitworth GB, Kress TL, Weissman JS *et al*: **A genetic interaction map of RNA-processing factors reveals links between Sem1/Dss1-containing complexes and mRNA export and splicing**. *Mol Cell* 2008, **32**(5):735-746.
212. Ho Y, Gruhler A, Heilbut A, Bader GD, Moore L, Adams S-L, Millar A, Taylor P, Bennett K, Boutilier K *et al*: **Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry**. *Nature* 2002, **415**(6868):180-183.
213. Gavin A-C, Bösche M, Krause R, Grandi P, Marzioch M, Bauer A, Schultz J, Rick JM, Michon A-M, Cruciat C-M *et al*: **Functional organization of the yeast proteome by systematic analysis of protein complexes**. *Nature* 2002, **415**(6868):141-147.
214. Gavin A-C, Aloy P, Grandi P, Krause R, Boesche M, Marzioch M, Rau C, Jensen LJ, Bastuck S, Dümpelfeld B *et al*: **Proteome survey reveals modularity of the yeast cell machinery**. *Nature* 2006, **440**(7084):631-636.
215. Li Z, Lee I, Moradi E, Hung N-J, Johnson AW, Marcotte EM: **Rational extension of the ribosome biogenesis pathway using network-guided genetics**. *PLoS Biol* 2009, **7**(10):e1000213.
216. Smirnova JB, Selley JN, Sanchez-Cabo F, Carroll K, Eddy AA, McCarthy JEG, Hubbard SJ, Pavitt GD, Grant CM, Ashe MP: **Global gene expression**

profiling reveals widespread yet distinctive translational responses to different eukaryotic translation initiation factor 2B-targeting stress pathways. *Molecular and Cellular Biology* 2005, **25**(21):9340-9349.
217. von der Haar T: **A quantitative estimation of the global translational activity in logarithmically growing yeast cells**. *BMC systems biology* 2008, **2**:87.
218. Beilharz TH, Preiss T: **Widespread use of poly(A) tail length control to accentuate expression of the yeast transcriptome**. *RNA* 2007, **13**(7):982-997.
219. Hogan DJ, Riordan DP, Gerber AP, Herschlag D, Brown PO: **Diverse RNA-binding proteins interact with functionally related sets of RNAs, suggesting an extensive regulatory system**. *PLoS Biol* 2008, **6**(10):e255.
220. Preiss T, Muckenthaler M, Hentze MW: **Poly(A)-tail-promoted translation in yeast: implications for translational control**. *RNA* 1998, **4**(11):1321-1331.
221. Cole JR, Olsson CL, Hershey JW, Grunberg-Manago M, Nomura M: **Feedback regulation of rRNA synthesis in Escherichia coli. Requirement for initiation factor IF2**. *Journal of Molecular Biology* 1987, **198**(3):383-392.
222. Yamagishi M, de Boer HA, Nomura M: **Feedback regulation of rRNA synthesis. A mutational alteration in the anti-Shine-Dalgarno region of the 16 S rRNA gene abolishes regulation**. *Journal of Molecular Biology* 1987, **198**(3):547-550.
223. Sabina J, Dover N, Templeton LJ, Smulski DR, Söll D, LaRossa RA: **Interfering with different steps of protein synthesis explored by transcriptional profiling of Escherichia coli K-12**. *J Bacteriol* 2003, **185**(20):6158-6170.
224. Ng W-L, Kazmierczak KM, Robertson GT, Gilmour R, Winkler ME: **Transcriptional regulation and signature patterns revealed by microarray analyses of Streptococcus pneumoniae R6 challenged with sublethal concentrations of translation inhibitors**. *J Bacteriol* 2003, **185**(1):359-370.
225. Park E-H, Zhang F, Warringer J, Sunnerhagen P, Hinnebusch AG: **Depletion of eIF4G from yeast cells narrows the range of translational efficiencies genome-wide**. *BMC genomics* 2011, **12**(1):68.
226. Berset C, Zurbriggen A, Djafarzadeh S, Altmann M, Trachsel H: **RNA-binding activity of translation initiation factor eIF4G1 from Saccharomyces cerevisiae**. *RNA* 2003, **9**(7):871-880.
227. Miura F, Kawaguchi N, Sese J, Toyoda A, Hattori M, Morishita S, Ito T: **A large-scale full-length cDNA analysis to explore the budding yeast transcriptome**. *Proc Natl Acad Sci USA* 2006, **103**(47):17846-17851.

228. Nagalakshmi U, Wang Z, Waern K, Shou C, Raha D, Gerstein M, Snyder M: **The transcriptional landscape of the yeast genome defined by RNA sequencing**. *Science* 2008, **320**(5881):1344-1349.
229. Yassour M, Kaplan T, Fraser HB, Levin JZ, Pfiffner J, Adiconis X, Schroth G, Luo S, Khrebtukova I, Gnirke A *et al*: **Ab initio construction of a eukaryotic transcriptome by massively parallel mRNA sequencing**. *Proc Natl Acad Sci USA* 2009, **106**(9):3264-3269.
230. Amrani N, Ghosh S, Mangus DA, Jacobson A: **Translation factors promote the formation of two states of the closed-loop mRNP**. *Nature* 2008, **453**(7199):1276-1280.
231. Fraser CS, Doudna JA: **Structural and mechanistic insights into hepatitis C viral translation initiation**. *Nat Rev Microbiol* 2007, **5**(1):29-38.
232. Iizuka N, Najita L, Franzusoff A, Sarnow P: **Cap-dependent and cap-independent translation by internal initiation of mRNAs in cell extracts prepared from Saccharomyces cerevisiae**. *Molecular and Cellular Biology* 1994, **14**(11):7322-7330.
233. Das S, Ott M, Yamane A, Venkatesan A, Gupta S, Dasgupta A: **Inhibition of internal entry site (IRES)-mediated translation by a small yeast RNA: a novel strategy to block hepatitis C virus protein synthesis**. *Front Biosci* 1998, **3**:D1241-1252.
234. Rosenfeld AB, Racaniello VR: **Hepatitis C virus internal ribosome entry site-dependent translation in Saccharomyces cerevisiae is independent of polypyrimidine tract-binding protein, poly(rC)-binding protein 2, and La protein**. *Journal of Virology* 2005, **79**(16):10126-10137.
235. Masek T, Vopalensky V, Horvath O, Vortelova L, Feketova Z, Pospisek M: **Hepatitis C virus internal ribosome entry site initiates protein synthesis at the authentic initiation codon in yeast**. *J Gen Virol* 2007, **88**(Pt 7):1992-2002.
236. Rosenfeld AB, Racaniello VR: **Components of the multifactor complex needed for internal initiation by the IRES of hepatitis C virus in Saccharomyces cerevisiae**. *RNA Biol* 2010, **7**(5).
237. Chen CR, Li YC, Chen J, Hou MC, Papadaki P, Chang EC: **Moe1, a conserved protein in Schizosaccharomyces pombe, interacts with a Ras effector, Scd1, to affect proper spindle formation**. *Proc Natl Acad Sci USA* 1999, **96**(2):517-522.
238. Yen HC, Chang EC: **Yin6, a fission yeast Int6 homolog, complexes with Moe1 and plays a role in chromosome segregation**. *Proc Natl Acad Sci USA* 2000, **97**(26):14370-14375.
239. Ray A, Bandyopadhyay A, Matsumoto T, Deng H, Maitra U: **Fission yeast translation initiation factor 3 subunit eIF3h is not essential for**

global translation initiation, but deletion of eif3h+ affects spore formation. *Yeast* 2008, **25**(11):809-823.
240. Crane R, Craig R, Murray R, Dunand-Sauthier I, Humphrey T, Norbury C: **A fission yeast homolog of Int-6, the mammalian oncoprotein and eIF3 subunit, induces drug resistance when overexpressed**. *Mol Biol Cell* 2000, **11**(11):3993-4003.
241. Heim R, Cubitt AB, Tsien RY: **Improved green fluorescence**. *Nature* 1995, **373**(6516):663-664.
242. Maundrell K: **Thiamine-repressible expression vectors pREP and pRIP for fission yeast**. *Gene* 1993, **123**(1):127-130.
243. Huh W-K, Falvo JV, Gerke LC, Carroll AS, Howson RW, Weissman JS, O'Shea EK: **Global analysis of protein localization in budding yeast**. *Nature* 2003, **425**(6959):686-691.
244. Livingstone CD, Barton GJ: **Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation**. *Comput Appl Biosci* 1993, **9**(6):745-756.
245. Humphrey T, Enoch T: **Sum1, a highly conserved WD-repeat protein, suppresses S-M checkpoint mutants and inhibits the osmotic stress cell cycle response in fission yeast**. *Genetics* 1998, **148**(4):1731-1742.
246. Zhan K, Vattem KM, Bauer BN, Dever TE, Chen J-J, Wek RC: **Phosphorylation of eukaryotic initiation factor 2 by heme-regulated inhibitor kinase-related protein kinases in Schizosaccharomyces pombe is important for fesistance to environmental stresses**. *Molecular and Cellular Biology* 2002, **22**(20):7134-7146.
247. Mateus C, Avery SV: **Destabilized green fluorescent protein for monitoring dynamic changes in yeast gene expression with flow cytometry**. *Yeast* 2000, **16**(14):1313-1323.
248. Iizuka N, Sarnow P: **Translation-competent extracts from Saccharomyces cerevisiae: effects of L-A RNA, 5' cap, and 3' poly(A) tail on translational efficiency of mRNAs**. *Methods* 1997, **11**(4):353-360.
249. Berry KE, Peng B, Koditek D, Beeman D, Pagratis N, Perry JK, Parrish J, Zhong W, Doudna JA, Shih I-h: **Optimized high-throughput screen for hepatitis C virus translation inhibitors**. *J Biomol Screen* 2011, **16**(2):211-220.