RP TU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Establishment of the Ribo-Seq method and Exploration of the Translatome in *Chlamydomonas reinhardtii*

vom Fachbereich Biologie

der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau

zur Verleihung des akademischen Grades Dr. rer. nat. genehmigte

Dissertation

vorgelegt von

Vincent Leon Gotsmann, M.Sc., geb. in Kirchheimbolanden

Mündliche Prüfung:	26.02.2026
Dekan:	Prof. Dr. Stefan Kins
Promotionskommisionsvorsitzender:	Prof. Dr. Timo Mühlhaus
Berichterstattende:	Prof. Dr. Felix Willmund
	Prof. Dr. Thorsten Stoeck

Kaiserslautern im Jahr 2026

D 386

Eigenständigkeitserklärung

Die vorliegende Dissertation wurde an der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau in der Arbeitsgruppe von Prof. Dr. Felix Willmund (zum jetzigen Zeitpunkt tätig an der Phillips-Universität Marburg) angefertigt. Mit dieser Erklärung bestätige ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und nur unter Zuhilfenahme der angegebenen Quellen und Hilfsmittel angefertigt habe. Die Arbeit wurde bisher weder im In- noch Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt. Die Bestimmungen der Promotionsordnung vom 10. Oktober 1996 sowie deren Änderung am 30. März 2001 des Fachbereichs Biologie sind mir bekannt.

Darlegung aller benutzten Hilfsmittel und Hilfestellungen

Die vorliegende Arbeit wurde mit Microsoft Word erstellt. Abbildungen wurden mit Microsoft Powerpoint, Inkscape, ChimeraX (Goddard et al., 2018) und Matplotlib (Hunter, 2007) erstellt und bearbeitet. Das Literaturverzeichnis und die Quellverweise wurden mithilfe von Mendeley Cite und Mendeley Reference Manager eingefügt. Zur Datenauswertung und statistischen Analyse wurden Microsoft Visual Studio Code, Anaconda, Jupyter (Kluyver et al., 2016), Python, R und R Studio genutzt. Genomische Annotationen wurden von Phytozome (Goodstein et al., 2012) und Ensembl Plants (Dyer et al., 2025) bezogen. Informationen zu einzelnen Proteinen wurden von UniProt (UniProt Consortium, 2025) bezogen. Zur Vorhersage der subzellulären Lokalisierung von Proteinen wurden PredAlgo (Tardif et al., 2012) und Target-P 2.0 (Almagro Armenteros et al., 2019) verwendet. Die Suche nach Fachpublikationen wurde mithilfe der Plattformen PubMed (www.pubmed.ncbi.nlm.nih.gov) und Google Scholar (www.scholar.google.de) durchgeführt.

Darlegung des Eigenanteil

Vincent Leon Gotsmann

Establishment of the Ribo-Seq method and Exploration of the Translatome in *Chlamydomonas reinhardtii*

Das Konzept des Forschungsprojekts wurde überwiegend von Prof. Dr. Felix Willmund entwickelt. Design und Planung von Experimenten sowie die Entwicklung der Ribo-Seq Protokolle wurde ebenfalls maßgeblich von Prof. Dr. Willmund unterstützt. Weiterhin waren Prof. Dr. Sophia Rudorf und Dr. Nadin Haase von der Leibniz Universität Hannover durch Diskussionsbeiträge bei der Konzeption bioinformatischer Analysen behilflich. Dr. Reimo Zoschke und Dr. Michael Kien Yin Ting vom Max-Planck-Institut für Molekulare Pflanzenphysiologie in Potsdam trugen zu der Planung und dem Design von Experimenten und der Entwicklung der Ribo-Seq Protokolle durch Informationsaustausch, Diskussion von Ergebnissen sowie durch die Bereitstellung eines Vergleichsprotokolls für Ribo-Seq in höheren Pflanzen bei. Die technische Umsetzung der Experimente, bioinformatische Auswertung von NGS-Daten sowie Entwicklung des Python Moduls „Ribo-Seq functions“ wurden von mir allein durchgeführt. Die Abbildungen Figure 3-15 und Supplementary Figure 6 sowie die zugrundeliegende Korrelationsanalyse wurden von Dr. Nadin Haase erstellt und zur Verfügung gestellt. Das Mischen und Sequenzieren von NGS-Bibliotheken sowie das Demultiplexen der gewonnen reads wurden von Jaro Schmitt, tätig an der Universität Heidelberg, durchgeführt.

Teilweise modifizierte Abbildungen wurden aus der folgenden Publikation übernommen:

Vincent Leon Gotsmann, Michael Kien Yin Ting, Nadin Haase, Sophia Rudorf, Reimo Zoschke, Felix Willmund (2024). Utilizing high-resolution ribosome profiling for the global investigation of gene expression in *Chlamydomonas*. The plant journal, 117(5), pp. 1614-1634. https://doi.org/10.1111/tpj.16577

Das Forschungsprojekt wurde im Rahmen des DFG-Projekts 437345987 sowie durch den Forschungsschwerpunkt BioComp der RPTU Kaiserslautern-Landau unterstützt. Die vorliegende Einschätzung über die erbrachte Leistung von Dritten wurde mit den genannten Personen einvernehmlich abgestimmt.

Content

1 Introduction

1.1 *Chlamydomonas reinhardtii* as a model organism

Chlamydomonas reinhardtii is a unicellular green alga commonly found in wet soils and freshwater ponds. A schematic depiction of *C. reinhardtii* is given in Figure 1-1. The cells are ovally shaped, approximately 10 µm in diameter and possesses two apical cilia. The alga's single cup-shaped chloroplast occupies up to 70% of a cell's volume and resides at the cells base (E. H. Harris, 2001). In contrast to higher plants, the cell wall of *C. reinhardtii* does not contain cellulose but consists primarily of hydroxyproline-rich glycoproteins (Poulhazan et al., 2024). A prominent feature of the cell's morphology is the eyespot, a small but bright yellowish

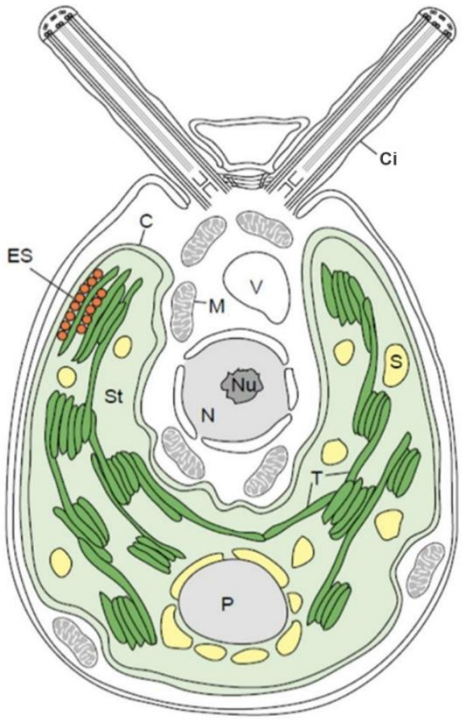


Figure 1-1: Schematic representation of a *C. reinhardtii* cell. Abbreviations: Ci: cilia, C: chloroplast, V: contractile vacuole, M: mitochondrion, ES: eye spot, St: stroma, T: thylakoid, P: pyrenoid, S: starch granule, N: nucleus, Nu: nucleolus (modified from Dent, Han and Niyogi, 2001)

organelle inside the chloroplast containing pigment-rich photoreceptor molecules that allow the cell to sense light and move phototactically (Seth et al., 2022). At the apex of the cell, close to the base of the cilia, the contractile vacuole is located - a pulsating vacuole continuously releasing water into the extracellular space to regulate the cell's osmotic state (Karin et al., 2014). Contrasting the usual textbook representation, the mitochondria of *C. reinhardtii* form a finely branched network spanning the surface of the chloroplast, most likely to optimize the exchange of metabolites between the two organelles (Findinier et al., 2024). Another prominent detail is the pyrenoid located at the base of the cell inside the chloroplast. This organelle is a

semi-crystalline accumulation of RuBisCO and other proteins involved in CO_2-fixation that is surrounded by starch sheaths and is permeated by fine tubular extrusions of the thylakoids. This spatial enrichment of CO_2-fixating enzymes represents a special strategy of green alga to increase inorganic carbon uptake known as “carbon concentrating mechanism” (CCM) (He et al., 2023). While the cells in laboratory environments are usually cultivated vegetatively via cell division, the organism is also able to reproduce sexually under environmentally unfavorable conditions. This requires the formation of gametes between cells of opposing mating types (which are desginated as “+” and “-“) and can be exploited to cross strains with different mutations (Dutcher, 1995). As a member of the green lineage, *C. reinhardtii* exhibits many similarities to higher plants, for example regarding the molecular structure of its photosynthesis machinery, gene expression principles and metabolism (E. H. Harris, 2001; Sasso et al., 2018). *C. reinhardtii* can be grown both phototrophically (relying on photosynthesis to harness energy) and heterotrophically providing acetate as carbon source (Heifetz et al., 2000). Under laboratory conditions, usually both light and acetate are provided to the alga. This “mixotrophic” condition drastically enhances cell division, shortening the doubling time of the organism from approximately 24 h under phototrophy to only 8 h (Stern et al., 2008). Due to this very short doubling time, the organism is a popular model for plant cell biology, allowing research at a much shorter time scale than in higher plants. Apart from pure plant cell biology, the alga is also a popular model for the study of cilia-related disorders. Due to the astonishing similarity of *C. reinhardtii*’s cilia to the cilia of human cells, for example of such as found in the respiratory tract, the organism is used to study the fine-structure and molecular processes taking place in cilia (Pan et al., 2005). In 2007, the nuclear genome was fully sequenced and continuously improved since then (Craig et al., 2023; Merchant et al., 2007). Due to their smaller size, it was possible to sequence the two organellar genomes of the mitochondrion and the chloroplast even earlier (Denovan-Wright et al., 1998; Maul et al., 2002). Both sequences were continuously improved and updated (Gallaher et al., 2018) and in the current version 6.1 of the nuclear genome assembly released in 2023, the two organellar genomes were added as extra chromosomes to provide one holistic genome version. In the current version, the nuclear genome consists of 17 chromosomes with a total length of roughly 114 megabases, harboring 16,801 annotated genes with 31,780 transcript variants. This makes the *C. reinhardtii* genome a comparably small but densely packed genome with approximately 60% of bases encoding genes. With a GC content of 64%, the genome can be considered GC-rich with some genes having a GC-content of up to 80%. The organellar genomes in contrast exhibit a GC content of only 35% (chloroplast) and 45% (mitochondrion), testifying their prokaryotic ancestry. The chloroplast genome is circular in structure and encodes 112 genes (including all duplicate genes) of which 72 are protein coding, 10 encode rRNAs, 29 encode tRNAs and 1 gene encodes a small RNA. The set of tRNAs found in the chloroplast is reduced as compared to

the nuclear genome, which has a set of 47 isodecoding tRNAs (Craig et al., 2023; Gallaher et al., 2018). However, this reduced set still contains one isoacceptor family for every amino acid. Despite the lost tRNAs, all codons can be served by a mechanism called “super-wobbling” that enables a chloroplast ribosome to accept near-cognate tRNAs when the first two bases of a codon are matched with the tRNA anticodon (Rogalski et al., 2008). The chloroplast genome contains two regions known as “large inverted repeats”, encoding rRNAs and the *psbA* gene (Turmel et al., 2017). As stated by the name, these regions resemble inverted copies of each other. Another interesting aspect is that the chloroplast encodes the photosystem I subunit *psaA* as a trans-spliced transcript. This transcript is split into three parts that are distributed over the genome without order and even over different strands (Lefebvre-Legendre et al., 2014). The whole chloroplast genome consists of 205,6 kilobases. The linearly structured mitochondrial genome is even shorter with only 15,8 kilobases and encodes 26 genes of which 8 encode proteins, 3 tRNAs and 15 rRNAs (Gallaher et al., 2018). In both organellar genomes, polycistronic arrangement of genes is widespread (Gallaher et al., 2021).

1.2 Origin and evolution of ribosomes

Ribosomes are nanoscale ribonucleoprotein particles consisting to large parts of RNA that are found in any cellular organism and which are essential parts of the gene expression machinery. In the central dogma of molecular biology, the ribosome is the machinery that translates the genetic information encrypted on nucleic acid molecules to polypeptide molecules (Crick, 1970). This task is accomplished by catalyzing the formation of peptide bonds between amino acids that are delivered to the ribosome in form of amino acylated tRNAs (Świderek et al., 2015). While genes are usually seen as the central orchestrators of life on the molecular level, it is often overlooked that the ribosome has a similarly central role by being the only biological entity that is able to transform the biochemical potential of a gene into biochemical action. Ribosomes inarguably belong to the evolutionarily oldest biomolecules known with their structure possibly being a testimony of the so-called “RNA World” – a hypothetical evolutionary state of cellular life that might have prevailed before DNA and proteins became the dominating types of biomolecules and biochemical reactions in early cells were presumably catalyzed mostly by structured RNAs known as ribozymes (Gilbert, 1986). Although the number of known ribozymes is relatively low (Scott, 2007), representants of this group are still found in organisms today and are partly of essential importance, like group I introns (Kruger et al., 1982; Zaug & Cech, 1986) or RNaseP (Guerrier-Takada et al., 1983). Notably, it was shown that fragments of the rRNA scaffold alone are able to produce short peptides from free amino acids in solution

without ribosomal proteins, proving that the catalytic activity of the ribosome is exerted by the RNA itself (Xu & Wang, 2021). For this reason, the ribosomal RNA can also be classified as a ribozyme, once more indicating its ancient origins. Using phylogenetic approaches, it was determined that the hypothetical last universal common ancestor ("LUCA") of today's life on earth should have had the full set of genes to form a functional translation machinery, indicating that the evolutionary origin of the ribosome even dates back into a pre-LUCA era before the evolution of life towards today's forms began (Bowman et al., 2020; J. K. Harris et al., 2003; Koonin, 2003). In general, the evolution of ribosomes and with that the ability to produce polypeptides based on a genetically encoded blueprint can be seen as a key event in primal evolution of today's life. Generally, ribosomes are divided into two types distinguished by their sedimentation coefficients: A prokaryotic 70S-type found in bacteria and archaea (Ramakrishnan, 2002; Y.-H. Wang et al., 2023) and an 80S-type found in eukaryotes (Wilson & Doudna Cate, 2012). Interestingly, ribosomes are also found in mitochondria (Greber & Ban, 2016) and chloroplasts (Bieri et al., 2017). According to the generally accepted endosymbiosis theory, these organelles originate from prokaryotes that endosymbiotically resided in early eukaryotic cells and were assimilated and reduced to semi-autonomous organelles during evolution. However, despite most genes of the organellar ancestors' genomes were eliminated or transferred to the nucleus of their hosts during evolution, they maintained a small genome and a gene expression machinery including ribosomes (Zimorski et al., 2014).

1.3 The molecular structure of ribosomes

Following a universal scheme, ribosomes consist of a small (30S for prokaryotic type, 40S for eukaryotic type) and a large subunit (50S for prokaryotic type and 60S for eukaryotic type) that both are formed by an RNA-scaffold (rRNA) decorated with ribosomal proteins (r-proteins) (Ramakrishnan, 2002; Wilson & Doudna Cate, 2012). While the RNA scaffold of the small subunit is always formed by a single rRNA (16S-rRNA in 70S ribosomes and 18S-rRNA in 80S ribosomes), the rRNA composition of the large subunit is more diverse across the domains of life and organellar origins (Melnikov et al., 2012). Ribosomes of mitochondria represent a special case since their composition seems to be much more diverse even across different species (Table 1-1).

Table 1-1: Composition of ribosomes from different domains and organelles (Ban et al., 2014; Greber & Ban, 2016; Londei, 2020; Londei & Ferreira-Cerca, 2021).

	bacteria	archaea	eukarya	chloroplast	mammalian mitochondria	yeast mitochondria
ribosome sedimentation	70S	70S	80S	70S	55S	74S
LSU sedimentation	50S	50S	60S	50S	39S	54S
LSU composition	23S, 5S	23S, 5S	26/28S, 5.8S, 5S	23S, 5S, 4.5S	16S	21S
SSU sedimentation	30S	30S	40S	30S	28S	37S
SSU composition	16S	16S	18S	16S	12S	15S
r-proteins	49 - 59	up to 68	78 - 80	up to 63	82	82

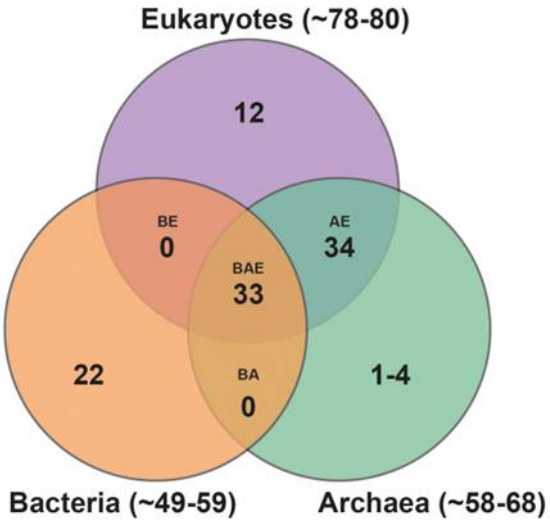


Figure 1-2: Venn diagram of universal, shared and exclusive r-proteins among domains of life. B: bacteria, E: eukaryotes, A: archaea. (Londei & Ferreira-Cerca, 2021)

While the large subunit of the ribosome contains the catalytic center of the particle (Noller et al., 1992), the small subunit exerts essential functions for the decoding of mRNA (Gornicki et al., 1984). rRNA sequences are usually highly conserved, qualifying them as optimal target sequences for phylogenetic comparisons (Bendich & McCarthy, 1970; Ludwig & Schleifer, 1994). In contrast to the rRNAs, the number and identity of r-proteins can differ much more across domains with only 33 r-proteins being universally present according to current knowledge (Figure 1-2) and many being domain-specific (Londei & Ferreira-Cerca, 2021). However, if present in different species, r-proteins usually exhibit a high degree of conservation as well (Korobeinikova et al., 2012). The rRNAs exhibit extensive structuration both on the secondary and the tertiary level, forming functional domains and providing specific binding sites for r-proteins (Herold & Nierhaus, 1987; Shajani et al., 2011; Traub & Nomura, 1968). During translation, the mRNA is threaded through a channel between the two subunits,

enabling the ribosome to translocate along the mRNA without losing contact (G. Yusupova et al., 2006; G. Z. Yusupova et al., 2001) (Figure 1-3). Moreover, the large subunit displays a large cavity above the mRNA channel that harbors the bodies of the three tRNAs bound to the active ribosome. This cavity contains the peptidyl transferase center of the particle which is located closely to the acceptor stems of the A- and P-site tRNAs to facilitate peptide bond formation (Shi & Moore, 2000; Simonović & Steitz, 2008). The three different binding sites for tRNAs are named A, P and E-site according to the state of the tRNA bound in them during

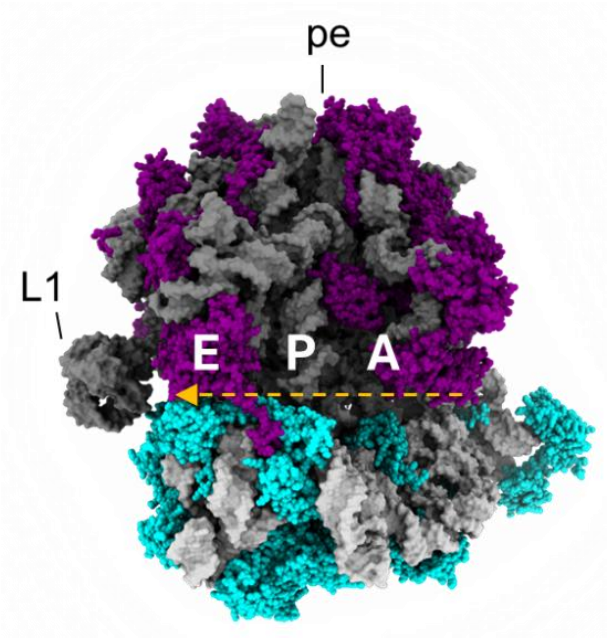


Figure 1-3: Structure of the chloroplast ribosome of *Spinacia oleracea*. The large subunit is represented by dark grey RNA residues and magenta-colored proteins while the small subunit is represented by light grey RNA residues and cyan proteins. Approximate positions of E-, P- and A-sites are indicated by respective white letters. L1 indicates the position of the L1 stalk (uL1c protein itself not included in the model). The approximate location of the polypeptide tunnel exit is indicated by pe. The orange arrow indicates the approximate path and direction of the mRNA through the ribosome.

translation (aminoacyl-, peptidyl- and exit-tRNA). The three sites are oriented from A- to E-site in 3'- to 5'-direction in respect to the mRNA (Rodnina et al., 2002). Roughly above the P-site, the polypeptide exit tunnel reaches through the large subunit up to the surface of the ribosome, where the polypeptide chain is threaded through towards the exterior during translation. Simultaneously, the electrostatic constraints of the tunnel force the chain to fold locally, representing the first line of co-translational protein folding (Samatova et al., 2024). Another prominent domain of the ribosome is the L1-stalk, located on the surface of the particle directly above the E-site. This stalk is named after the r-protein uL1 which resides at this structure and swings in and out of the E-site during translation, suspectedly aiding in the release of deacylated tRNA from the E-site (Trabuco et al., 2010).

Biochemistry typically encompasses four levels of structure that are considered in protein characterization. The primary structure is represented by a protein's pure amino acid

sequence. The secondary structure is represented by the local fold of protein domains along the polypeptide chain. The tertiary structure describes how these domains are arranged in three-dimensional space and the quaternary structure how multiple folded proteins are arranged to form functional complexes (Anfinsen, 1973). However, considering the whole proteome of a cell, compartmentalization of proteins and their distribution - their quinary structure - becomes important to cellular functions (Cohen & Pielak, 2017). An aspect of the ribosome often overlooked is that the particle is densely packed with more than 4,000 negative charges due to the phosphate residues of the rRNA scaffold (Watson et al., 2020). Due to the electrostatic effects of such a high charge density and the typically high abundance of ribosomes in the cell, they passively influence charged molecules in their surroundings and participate in orchestrating the quinary structure around them (Breindel et al., 2020; DeMott et al., 2017; Schavemaker et al., 2017). Crosslinking studies prove that ribosomes closely contact an astonishing number of proteins not involved in translation, supporting the hypothesis that the ribosome is also important as a condensation point for the arrangement of biomolecules (Simsek et al., 2017; Westrich et al., 2021).

1.4 The translation process

In general, the process of translation is divided into the three steps of initiation, elongation and termination that universally apply to all species of ribosomes. However, the exact mechanisms of the three stages and the co-factors involved partly differ drastically between prokaryotes and eukaryotes (Figure 1-4).

During initiation, the small subunit binds a transcript molecule and after positioning the P-site on the start codon of the encoded open reading frame (ORF), the large subunit joins, and translation of the mRNA starts. This is achieved very differently for 70S and 80S ribosomes. For prokaryotes, the model of Shine-Dalgarno dependent initiation is best studied and often seen as the bona fide mode of initiation in bacteria (Jiong et al., 2002; Shine & Dalgarno, 1974). In this model, the small subunit of the ribosome forms a labile pre-initiation complex (30S-PIC) with the initiation factors IF1, IF2, IF3 and a tRNA carrying a formylated methionine molecule (fMet-tRNA) (Gualerzi & Pon, 2015). The 30S-PIC binds a transcript molecule via base interaction with the Shine-Dalgarno Sequence (SD), a conserved sequence (“AGGAGG” in *Escherichia coli*) of nucleotides upstream of the start codon of an ORF. This sequence is complementary to the anti-Shine-Dalgarno sequence (antiSD) “CCUCCU” located at the 3’-end of the 16S-rRNA (Amin et al., 2018; H. Chen et al., 1994), which resides at the surface of the small subunit neighboring the E-site of the ribosome in the Shine-Dalgarno cleft (G. Z.

Yusupova et al., 2001). Base pairing of SD and anti-SD bring the 30S-PIC and the transcript to be translated into close contact with each other, enabling the 30S-PIC to bind the mRNA and positioning the start codon at the P-site of the 30S-PIC forming the 30S initiation complex (30S-IC) (Korostelev et al., 2007). Afterwards, binding of the large ribosomal subunit and release of IF1, IF2 and IF3 from the complex forms the 70S initiation complex which enters the elongation stage upon binding of an aminoacyl-tRNA at the A-site (Fabbretti et al., 2007; Goyal et al., 2015). Although this is the most referred model for initiation in prokaryotes, it should be noted that the relevance of SD-dependent initiation among bacteria is variable (Estrada et al., 2024; Nakagawa et al., 2017).

For eukaryotes, initiation differs dramatically from the SD-dependent model for bacteria and involves many more co-factors. To achieve initiation, the 43S-preinitiation complex, consisting of the 40S subunit, eIF2, Met-tRNA, eIF3, eIF1, eIF5 and eIF1A binds to the 5'-terminal N7-methylguanosin group (5'-cap) of eukaryotic transcripts under assistance of eIF4A, eIF4B and eIF4F, which unfold the secondary structure of the 5'-terminus of the transcript (Hinnebusch, 2014; Mishra et al., 2020). The 43S-PIC then scans the transcript towards the 3'-terminus until the start codon of an ORF is recognized. To discriminate a correct start codon against AUG triplets within the 5'-UTR, start codons in the context of a "Kozak sequence", a species-dependent sequence motif, are recognized by the 43S-PIC (Hamilton et al., 1987; Kozak, 1986; Lütcke et al., 1987). After start codon recognition, the large ribosomal subunit together with eIF5B joins the complex upon release of all previous initiation factors except for eIF1A. Afterwards, eIF1A and eIF5B are released as well, forming the final 80S-initiation complex (Hinnebusch, 2014).

After the ribosomes have been fully assembled in a complex with mRNA with the start codon of an ORF bound to the P-site, the ribosome enters the elongation stage. Elongation is generally very similar in eukaryotes and prokaryotes with the factors involved being homologous. Elongation can be divided into a cycle of three phases that repeat for each codon of an ORF: decoding, peptide bond formation and translocation. The decoding of the codon residing in the A-site of the ribosome is facilitated by the delivery of aminoacyl tRNAs to the ribosome by elongation factors EF-Tu in prokaryotes or eEF1A in eukaryotes (Dever et al., 2018; Rodnina et al., 2017). When a codon is recognized by the anticodon of the tRNA, the elongation factor is released from the complex, leaving behind the amino-acylated tRNA. Afterwards, peptide bond formation is promoted by factor EF-P (prokaryotic) or eIF5A (eukaryotic) binding to the E-site of the ribosome (Doerfel et al., 2015; Gutierrez et al., 2013). Peptide bond formation itself is catalyzed by the peptidyl transferase center (PTC) of the ribosome which transfers the nascent polypeptide chain from the peptidyl tRNA bound at the P-site to the amino acid of the aminoacyl tRNA bound to the A-site. In detail, the chain is

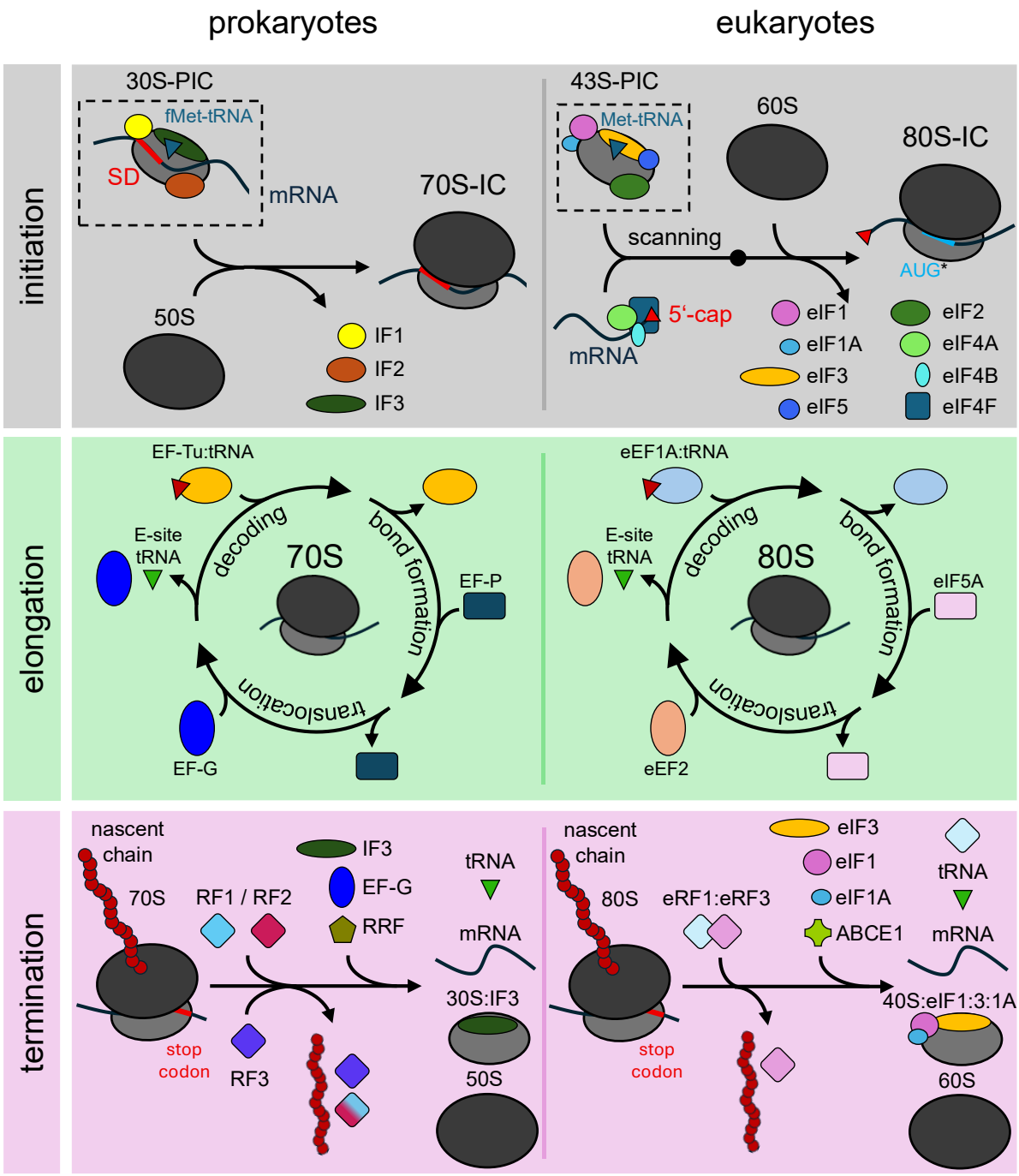


Figure 1-4: Schemes of translation in prokaryotes and eukaryotes. Top panel illustrates translation initiation in prokaryotes (left) and eukaryotes (right) SD indicates Shine-Dalgarno sequence, AUG* indicates AUG in context of a Kozak sequence. The black dot on the arrow on the right side indicates the point of start codon recognition while scanning. Middle panel illustrates the cycle of tRNA decoding, peptide bond formation and translocation during the elongation stage. Lower panel illustrates the prokaryotic and eukaryotic modes of termination and ribosome recycling, respectively. Factors and complexes are indicated by colored shapes and given names. Complexes of factors are named A:B. Prokaryotic ribosomal subunits are indicated by 30S and 50S while complete ribosomes are indicated by 70S. Eukaryotic ribosomal subunits and ribosomes are indicated analogously by 40S, 60S and 80S. PIC indicates pre-initiation complexes and IC indicates fully assembled initiation complexes. 5'-cap indicates the 5'-terminal m^7G-cap of eukaryotic mRNAs.

elongated through a nucleophilic attack of the aminoacyl tRNA's α-amino group on the carbonyl carbon of the ester bond between the C-terminal chain residue and the ribose residue of the 3'-terminal base of the P-site tRNA (Rodnina & Wintermeyer, 2003). Afterwards, structural rearrangements rotate the subunits of the ribosome against each other to bring the tRNAs into

hybrid states between A/P- or P/E-sites, respectively (Budkevich et al., 2011; Moazed & Noller, 1989). Next, elongation factor EF-G or in eukaryotes eEF2 binds to the A-site to induce the final translocation from P- to E- and A- to P-site, accompanied by backwards rotation of the subunits and the movement towards the 3'-end of the translated mRNA (Ling & Ermolenko, 2016; Noller et al., 2017). Due to the characteristic opposing rotations of the ribosomal subunits, the translocation is often described as "ratchet-like" movement along the mRNA.

When a ribosome incorporates a stop codon (UAA, UGA and UAG) into its A-site, termination begins. Instead of tRNAs, the stop codons are recognized by ribosome release factors. In prokaryotes, these are RF1 or RF2, depending on the exact base triplet of the stop codon. The respective release factor binds to the A-site and hydrolyzes the ester bond between the nascent chain and the peptidyl tRNA. Next, RF3 binds to the ribosome, releasing RF1/2 and the nascent protein from the complex (Rodnina, 2018). In eukaryotes, termination is facilitated by only one complex of eRF1 and eRF3 that does both, recognizing the stop codon and promoting peptide release (Hellen, 2018). After the nascent chain is released, the prokaryotic ribosome is recycled by subunit dissociation through the concerted action of ribosome recycling factor (RRF) and EF-G. After dissociation, the small subunit is still bound to both the tRNA and the transcript. While the transcript is released spontaneously, the tRNA is released by binding of IF3 upon recruitment of the subunit to the next pre-initiation complex (Rodnina, 2018). In eukaryotes, ribosome recycling relies on the action of eRF1 together with ABCE1. The release of mRNA and deacylated tRNA from the 40S subunit is then facilitated by eIF1, eIF1A and eIF3 (Hellen, 2018).

1.5 Synthesis and import of chloroplast proteins

In general, the translation machinery of chloroplasts resembles the prokaryotic scheme (Ban et al., 2014; Yamaguchi et al., 2000; Yamaguchi & Subramanian, 2000). However, while the overall circumstances are similar, the chloroplast translation machinery has developed certain characteristics over the course of the organelle's evolution. First, while translation is temporally and spatially coupled to transcription in procaryotes, this coupling is thought to be much less pronounced in chloroplasts, if persistent at all (Zoschke & Bock, 2018). While the chloroplast ribosome is of the 70S type and generally very similar to the bacterial 70S ribosome, seven plastid-specific ribosomal proteins are known (PSRPs) (Beligni et al., 2004; Yamaguchi & Subramanian, 2003). Currently, the exact roles of these proteins are still under investigation. Although it is proposed that these do not represent pure structural constituents of the ribosome, evidence on their roles is scarce. At least in case of PSRP1 a regulatory role could not be

confirmed yet (Sharma et al., 2010; Swift et al., 2020). Just like its bacterial relatives, the chloroplast ribosome has a fully preserved anti-Shine-Dalgarno sequence at the 16S rRNAs 3'-end (Bieri et al., 2017). However, the relevance of this sequence feature is under discussion since only part of chloroplast transcripts possess SD sequences and if they do, they are often degenerated (Wei & Xia, 2019). So far, studies on the subject have presented conflicting outcomes between species (Fargo et al., 1998; Scharff et al., 2017). However, in some cases mutation of SD-sequences in chloroplast transcripts has been shown to negatively affect these transcripts' expression on the translational level (Scharff et al., 2017). It is very likely that translation initiation in chloroplasts depends on a variety of different factors whose impact on the translation of specific transcripts may be variable according to environmental conditions, development stage and other influences rather than on the presence or absence of a SD-sequence alone. Due to the low number of chloroplast-encoded proteins (usually less than 100), it is to assume that the chloroplast gene expression machinery exerts a high degree of adaptation to the synthesis of these specific proteins, including usually a high number of exceptionally short membrane proteins which are mostly subunits of the protein complexes involved in photosynthesis (Gallaher et al., 2018; Gao et al., 2016; Sato et al., 1999; She et al., 2022) . However, not all subunits of these complexes are encoded on the chloroplast genome. In fact, most chloroplast localized proteins (~3,000) are encoded on the nuclear genome and have to be imported through the chloroplast envelopes and, if necessary, be assembled with chloroplast-encoded complex partners to fulfill their function (Sun & Jarvis, 2023). It is assumed that protein import into the chloroplast is mainly post-translationally facilitated by keeping the polypeptide in an unfolded state after synthesis in the cytosol via complexion with chaperones (Fellerer et al., 2011). Subsequent targeting to the chloroplast envelope is facilitated by N-terminal transit sequences that are recognized by proteins of the chloroplast outer envelope translocon (TOC) (Jarvis et al., 1998; Bauer et al., 2000), which together with the chloroplast inner envelope translocon (TIC) forms the TOC-TIC pore complex spanning both envelopes (Jin et al., 2022). At this complex, the polypeptide chain is threaded through the pore and folds within the stroma into its native conformation or is directly transported to the thylakoid membrane in case of thylakoid proteins. This mechanism was especially well characterized for light harvesting complex proteins (LHCPs), which are received by the chaperones cpSRP54 and cpSRP43 at the inner side of the pore complex and are recruited to the thylakoids for insertion into the membrane and incorporation into the photosystems (Ziehe et al., 2018). cpSRP54 is a homolog of the cytosolic SRP54 which is involved in protein targeting to the endoplasmatic reticulum and the prokaryotic signal recognition particle (SRP). In bacteria, this complex attaches co-translationally to the ribosome at the polypeptide-exit channel upon emergence of a specific signal sequence in the nascent polypeptide chain. Afterwards, elongation is paused and the ribosome is recruited to the outer

membrane, where the SRP interacts with a pore complex that receives the nascent chain and inserts it into the membrane upon translation (Kellogg et al., 2021). In the model plant *Arabidposis thaliana* at least cpSRP54 has been found to interact with chloroplast ribosomes upon translation of thylakoid membrane proteins by pausing the translating ribosome and recruiting it to the thylakoid membrane (Hristou et al., 2019). For cpSRP43, which is exclusively found in chloroplasts (Klimyuk et al., 1999), no evidence for a co-translational function has been presented yet. While the post-translational import of chloroplast proteins is relatively well studied and was long assumed to be the canonical import mode for all chloroplast proteins, it was just recently found that nuclear encoded chloroplast proteins are also co-translationally imported into the plastid by cytoplasmic ribosomes attached to the chloroplast outer envelope (Sun et al., 2024).

1.6 Approaches to analyze translation

Classical approaches to analyze translation are usually based either on pulse-labeling or polysome analysis. Pulse-labeling is used to determine the amount of protein that was produced in a certain timespan while polysome profiling approaches quantify the amount of transcript that can be purified from the polysomal fraction of a cell lysate (Bostrom et al., 1986; Chambers & Ness, 1997). For pulse labeling, cells are supplemented for a defined time with radioactively labeled amino acids which are incorporated into newly synthesized protein molecules. Afterwards, the labeled amino acids are removed from the media to stop the uptake of radioactivity and samples are taken in defined intervals. Subsequently, radiolabeled proteins can be visualized by SDS-PAGE and autoradiography, allowing for determination of the turnover rate of a protein species by following the decrease of radioactivity over time (Fritzsche & Springer, 2014). Since protein turnover is defined by protein synthesis and degradation, the protein synthesis rate can be inferred from the turnover rate if a steady state is assumed (Bostrom et al., 1986). However, the method suffers from the limited resolution of gel electrophoresis and the masking of proteins with similar molecular weight. In the modern pcSILAC (pulse-chase stable isotope labeling by amino acids in cell culture) approach, labeling with non-radioactive heavy isotopes in combination with proteomics can be used to circumvent these problems and obtain a higher resolution in the read-out. This method represents an adaptation of the standard SILAC approach to specifically determine protein synthesis and degradation rates (X. Chen et al., 2015; Fierro-Monti et al., 2013). However, while pulse-labeling approaches in general are well suited to quantify protein synthesis rates, they do not provide information about the mechanics of translation and their dynamics.

Polysome profiling in contrast focuses on the determination of the polysome-bound fraction of a specific transcript. Upon sampling, cells are treated with elongation inhibitors that stop actively translating ribosomes and fixate them on the translated transcript, resulting in inactive polysome-complexes that can be enriched by ultracentrifugation either through a sucrose density gradient or a sucrose cushion. Afterwards, specific transcripts of interest are detected and quantified in the polysomal fraction(s) via northern blotting (Chassé et al., 2017). Just like the pulse-labeling approach was improved by modern MS techniques, polysome profiling was modernized with the advent of high-throughput sequencing techniques, allowing researchers to quantify polysome-enriched transcripts genome-wide (Pereira et al., 2018). However, while this technique provides an accurate measure of the fraction of each transcript that is bound by ribosomes, it allows only for very limited analysis of translational dynamics in the sense of a transcript's propensity for polysomal attachment.

While both methods provide a certain measure for translational activity, they both suffer from the inability to provide direct observations on the mechanics and dynamics of the translation process.

1.7 The development of Ribo-Seq

With the advent of Next Generation Sequencing techniques, analysis of whole genomes and transcriptomes became a feasible task for biologists. Transcriptomics became a new popular field of research complementing genomics and proteomics, closing a gap in the palette of omics-techniques. However, transcriptomics is only partly able to explain changes that occur on proteomic level. Instead, it became evident that a high degree of regulation must occur at the post-transcriptional level, rendering translation a much more complex process than anticipated (Baek et al., 2008; Selbach et al., 2008; Sonenberg & Hinnebusch, 2007). In 2009, the Ribo-Seq technique was published by combining classical ribosome profiling approaches with deep sequencing, allowing not only to approximate translation rates of transcripts but also to observe the dynamics of translation for every transcript individually (Ingolia et al., 2009). The technique is schematically depicted in Figure 1-5. It is based on incubating cells with high concentrations of translation elongation inhibitors (usually cycloheximide for eukaryotic ribosomes and chloramphenicol for prokaryotic ribosomes) causing actively translating ribosomes to arrest while remaining attached to the mRNA. Afterwards, the ribosomes are enriched via ultracentrifugation and digested with carefully titrated amounts of nucleases (Gerashchenko & Gladyshev, 2017). As a result, the parts of mRNA not shielded by ribosomes are degraded while the ribosomes remain largely intact due to extensive structuration of their

rRNA. While the use of RNAseI is popular in eukaryotic protocols, micrococcal MNase became a popular nuclease for Ribo-Seq in bacteria since RNaseI was found to be inhibited by bacterial rRNA (Becker et al., 2013; Mohammad et al., 2019). After nuclease digest, the ribosome footprints or ribosome protected fragments (RPFs) are extracted and further purified via common RNA-extraction procedures followed by size selection by electrophoresis. The general length of RPFs range around 30 nucleotides with prokaryotes usually exhibiting slightly smaller lengths (Glaub et al., 2020; Ingolia et al., 2009). RPFs are then used as input for NGS library preparation and sequenced analogously to typical RNA-Seq samples. Ribo-Seq experiments are not restricted to the quantification of translatome (the entirety of translated transcripts, analogous to the transcriptome) but also contain information about the dynamics of translation for each transcript in the form of ribosome profiles which display the distribution of RPFs across an ORF. For example, if ribosomes pause at a specific site during translation of an ORF, possibly due to a co-translational event, the chance of sampling ribosomes at this specific site increases, leading to an accumulation of RPFs at the respective site of the ORFs ribosome profile (Flanagan et al., 2022). Such information provides deep insight into the biogenesis of specific proteins and can potentially help to identify sites of co-translational folding or membrane insertion events. Metagene analyses of Ribo-Seq data across many transcripts at once provide a possibility to study general aspects of translation or to assess the quality of the data. One example of such an analysis is the determination of the “triplet periodicity”. Since ribosomes move triplet-wise along an ORF and since the time needed to decode a triplet and form a peptide bond is longer than the time needed to translocate (Rodnina & Wintermeyer, 2016), it is more likely to sample ribosomes with their P-site bound to a base triplet in-frame with the translated ORF than out of frame. By mapping the P-sites of RPFs on the metagene level against their frame, a clear periodic pattern should appear with most RPFs having in-frame P-sites. To identify the P-site codon of an RPF, the 5’-offset - the distance between the ribosome’s P-site and the 5’-end of RPFs - must be determined first. This is typically done by exploiting the fact that initiating ribosomes have the start codon bound at their P-sites and are abundant due to initiation being a slow process (Shah et al., 2013). By mapping the distance between the start codon and their 5’-ends for all RPFs that cover a start codon, the 5’-offset should become evident for each RPF length species separately as the most frequently occurring offset (Dunn & Weissman, 2016; Popa et al., 2016).

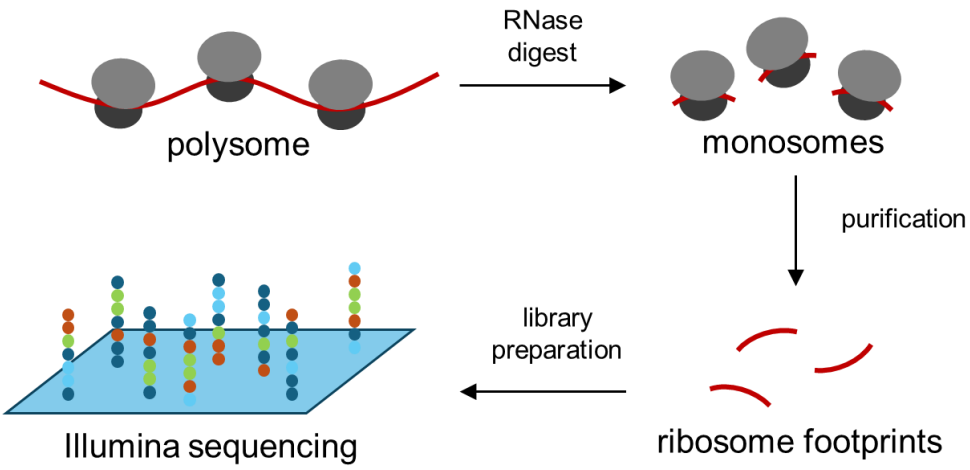


Figure 1-5: Schematic representation of a standard Ribo-Seq workflow. After treatment with elongation inhibitors, polysomes are digested into monosomes and RPFs are subsequently purified. NGS-libaries are prepared from purified RPFs and are sequenced and analyzed.

The Ribo-Seq technique was already adapted to analyze several specific aspects of translation (Becker et al., 2013; Bertolini et al., 2021; Oh et al., 2011; VanInsberghe et al., 2021). One example of such adaptations is the selective ribosome profiling (seRP) approach, coupling immunoprecipitation of specific translation co-factors with Ribo-Seq to enrich ribosomes bound to respective co-factors. In this way, transcripts that depend on this factor for their translation can be identified and studied as well as the mechanics of the ribosome-co-factor interaction (Oh et al., 2011; Schibich et al., 2016).

1.8 Aims of this study

This study aims to establish a working protocol for Ribo-Seq in *C. reinhardtii* that can further be utilized as a foundation to develop a selective ribosome profiling workflow. Furthermore, the capabilities and limitations of the technique should be evaluated by developing custom bioinformatic analysis workflows that allow the user to inspect Ribo-Seq data sets in greatest detail. This study aims to test the technique in a case-study of mixotrophic *C. reinhardtii* cultures to explore the translatome of the alga with special emphasis on the chloroplast translatome. Furthermore, a comparative analysis of an inducible knock-down mutant of ribosomal protein uS11c was carried out to find chloroplast transcripts directly depending on the protein for translation. Additionally, a first approach of a selective ribosome profiling experiment is supposed to yield information about the targets of cpSRP54's putative co-translational action in the chloroplast of *C. reinhardtii*.

2 Materials & Methods

2.1 Materials

2.1.1 Software used in this work

Table 2-1: Software used in this work.

Purpose	Software
ECL-detection	LabImage 1D
Microscopy	Olympus CellSense Dimension
Image processing	ImageJ 1.48v Microsoft PowerPoint version 2407 InkScape 1.2.2
Statistics, data processing and presentation	Microsoft Excel version 2407 Python 3.9 Microsoft Visual Studio Code 1.78.2
Word processing	Microsoft Word version 2407 Elsevier Mendeley 1.17.10
Programming packages and bioinformatic tools	Cutadapt 3.2 (Martin, 2011) STAR 2.7.7a (Dobin et al., 2013) FastQC 0.12.1 (Andrews et al., 2012) UMItools 1.1.1 (Smith et al., 2017) Jupyter 3.3.2 (Kluyver et al., 2016) Scipy 1.5.3 (Virtanen, Gommers, Oliphant, Haberland, Reddy, et al., 2020) Statsmodels.0.13.2 (Seabold & Perktold, 2010) DESeq2 1.48.2 (Love et al., 2014) Pandas 1.4.3 (McKinney, 2010) Numpy 1.20.3 (C. R. Harris et al., 2020) Scikit-learn 1.2.0 (Pedregosa et al., 2011) Ribo-Seq functions (Gotsmann et al., 2024) Anaconda 2-2.4.0. BCL2fastq 2.20.0.422 Target-P 2.0 webserver (Almagro Armenteros et al., 2019)

Purpose	Software
	PredAlgo webserver (Tardif et al., 2012) Pysam 0.16.0.1 (https://github.com/pysam-developers/pysam) Matplotlib 3.3.4 (Hunter, 2007) Samtools 1.3.1 (Danecek et al., 2021) Custom scripts (see section 0 to 8.5)

2.1.2 Devices used in this work

Table 2-2: Devices used in this work.

Type of device	Name and manufacturer
Centrifuges	Centrifuge 5417 R (Eppendorf AG, Hamburg) Multifuge X3R (Thermo Fisher Scientific, Dreieich) Ultracentrifuge L8-70M (Beckmann Coulter GmbH, Krefeld) Avanti J-26S XP (Beckmann Coulter GmbH, Krefeld) Tabletop Ultracentrifuge Optima MAX-XP (Beckmann Coulter GmbH, Krefeld) Sorvall MX 150 Plus (Thermo Fisher Scientific, Dreieich)
Particle counter	Z2 Coulter® Particle Count and Size Analyzer (Beckmann Coulter GmbH, Krefeld)
Thermomixer	Thermomixer comfort (Eppendorf AG, Hamburg)
SDS-PAGE	Mini-Protein-Electrophoresis-System (Bio-Rad Laboratories GmbH, München) Midi- und Maxi Protein-Electrophoresis-System (Peqlab Biotechnologie GmbH, Erlangen)
Western Blot	PerfectBlueTM „Semi-Dry“-Elektroblotter (Peqlab Biotechnologie GmbH, Erlangen)
Power Supply	PowerPac 300 (Bio-Rad Laboratories GmbH, München)
Chemiluminescence- / Fluorescence system	ChemoStar PC ECL & Fluorescence Imager (iNTAS, Göttingen)
Microscope	BX53F (Olympus GmbH, Hamburg)
Thermo-Cycler	Tpersonal 48 (Biometra, Göttingen)
Scales	AE160 (Mettler, Gießen) PJ3000 (Mettler, Gießen)
Fragment Analyzer	BiOptic Qsep1 (NIPPON Genetics EUROPE, Düren)
Microvolume UV-Vis Spectrophotometer	NanoDrop 2000 (Thermo Fisher Scientific GmbH, Dreieich)
Microvolume Fluorometer	Qubit 4 (Thermo Fisher Scientific GmbH, Dreieich)

Type of device	Name and manufacturer
Ball Mill	MM500 Control (Retsch GmbH, Haan)
LED table	Blue/Green-Transilluminator (iNTAS, Göttingen)
Ultracentrifugation tubes	1 mL Open-Thickwall PP tubes, 11x34 mm (Beckmann Coulter GmbH, Krefeld)
Ultracentrifuge rotors	MLA-130 Fixed Angle Rotor (Beckmann Coulter GmbH, Krefeld) Type 90 Ti Fixed-Angle Titanium Rotor (Beckmann Coulter GmbH, Krefeld) S150-AT Fixed Angle Rotor (Thermo Fisher Scientific GmbH, Dreieich)
Bioreactor	BioBench flat panel bioreactor (Biostream International, Doetichem (NL))
Sequencer	Illumina NextSeq 550 Sequencing System (Illumina, Inc., San Diego (USA)))

2.1.3 Chemicals and reagents used in this work

Chemicals used in this work and not listed explicitly in the tables below (mainly applying to salts, bases and acids) were sourced from suppliers Roth and Merck in technical or analytical grade.

Table 2-3: Enzymes, speciality chemicals and kits used in this work.

Reagent	Manufacturer	Manufacturer article number
Ambion RNAse I	Thermo Fisher Scientific	AM2295
TURBO DNase		AM2239
SUPERaseIn RNase Inhibitor		AM2696
SYBR Gold nucleic acid stain		S11494
GlycoBlue Co-precipitant		AM9516
TRIzol reagent		15596026
Ultra Low Range DNA Ladder		10597012
TrackIt Cyan/Yellow Loading Buffer		10482035
Dynabeads MyOne Streptavidin C1		65001
DMP		21667
DSP		22586
Cycloheximide		J66901.03
Chloramphenicol		B20841.14
Qubit RNA High Sensitivity Kit		Q32852
Qubit RNA Extended Range Kit		Q33223
Qubit RNA Broad Range Kit		Q10210

Reagent	Manufacturer	Manufacturer article number
Qubit microRNA Kit		Q32880
Protein A Resin - Amintra	abcam	ab270308
NP-40 substitute	Merck Millipore	492018
Prestained Marker for Small RNA Plus	DynaMarker	DM253S (discontinued)
RNase H	New England Biolabs	M0297S
T4 Polynucleotide Kinase		M0201S
Monarch Spin RNA Cleanup Kit (10 µg)		T2030L
NucleoSpin RNA Plant and Fungi Kit	Macherey Nagel	REF 740120.250
Costar Spin-X centrifuge tube filters	Corning	8162
NEXTFLEX Small RNA Sequencing Kit v3	Revvity (former Perkin Elmer)	NOVA-5132-31
Qsep1 S2 Fragment Analyzer Cartridge Kit	Nippon Genetics	C105101
Zymo-Seq RiboFree Total RNA Library Kit	Zymo Research	R3000
NextSeq 500/550 High output Kit v2.5 (75 Cycles)	Illumina	20024906
MaXtract High Density Tubes	Qiagen	129046
Qsep1 20 bp & 1000 bp Alignment marker	Nippon Genetics	C109100

Table 2-4: Biotinylated oligonucleotide mix used for rRNA depletion. [Btn] indicates the biotin tag.

Oligo	Sequence	Final Concentration
FW733	[Btn]AATATGCGTTCAAAGATTCGATGATTCACG	58.8 µM
FW734	[Btn]TAGCTCTAGAATTACTACGGTTATCCGAGTA	11.8 µM
FW735	[Btn]TACCCGACGCTGAGGCAGACATGCTCTTGG	11.8 µM
FW736	[Btn]GATTCGTGAAGTTATCATGATTCACCGCA	5.9 µM
FW737	[Btn]ACGGGATGAATCTCAGTGGATCGTAGCA	5.9 µM
FW738	[Btn]CGATCTAGCCGTCTTAGAGCTAGAAGCAGG	5.9 µM
Total:	-	100 µM

2.1.4 Buffers used in this work

This section describes the composition of buffers used in this work and gives specific details on their preparation, use and handling.

Table 2-5: Buffers used in this work in the Ribo-Seq workflow.

Compound	Final concentration
2 x TKM buffer Solution was sterile filtered after preparation, then aliquoted into 50 mL tubes and stored frozen until use.	
Tris-acetate (stock adjusted to pH 8.0)	100 mM
KCl	300 mM
$MgCl_2$	20 mM
RNase-free H_2O	(fill up to desired volume)
TKM+ buffer Solution was sterile filtered after preparation, then aliquoted into 50 mL tubes and stored frozen until use.	
2 x TKM buffer	1 x
Chloramphenicol	100 µg/mL
Cycloheximide	100 µg/mL
RNase-free H_2O	(fill up to desired volume)
PBS buffer Solution was autoclaved for long-term storage.	
NaCl	137 mM
KCl	2.7 mM
Na_2HPO_4	10 mM
KH_2PO_4	1.8 mM
Phosphate buffer Solution was freshly prepared from autoclaved stock solutions and discarded after use. If necessary, pH was adjusted to 7 by using HCl or NaOH.	
Na_2HPO_4	57.8 mM
NaH_2PO_4	42.2 mM

2 x base buffer 2 x base buffer was used as base for TKM-D, TKM-T, 2x Lysis buffer, sucrose cushion and Ribosome buffer. Solution was sterile filtered after preparation, then aliquoted into 50 mL tubes and stored frozen until use.	
Tris-acetate (stock adjusted to pH 8.0) (HEPES, pH 8.0 in case of selective ribosome profiling)	100 mM
KCl	300 mM
$MgCl_2$	20 mM
Chloramphenicol	200 µg/mL
Cycloheximide	200 µg/mL
DTT	2 mM
RNase-free H_2O	(fill up to desired volume)
TKM-T buffer Solution was sterile filtered after preparation, then aliquoted into 50 mL tubes and stored frozen until use.	
2 x base buffer	1 x
20% Tween-20	0.05%
RNase-free H_2O	(fill up to desired volume)
TKM-D buffer Solution was sterile filtered after preparation, then aliquoted into 50 mL portions and stored frozen until use.	
2 x base buffer	1 x
RNase-free H_2O	(fill up to desired volume)
Ribosome buffer Solution was sterile filtered after preparation, then aliquoted into 2 mL tubes and stored frozen until use.	
2 x base buffer	1 x
Roche Protease Inhibitor Cocktail	4 x
PMSF	1 mM
Invitrogen TURBO DNase	0.134 U/µL
Triton X-100	0.1%
RNase-free H_2O	(fill up to desired volume)

2 x Lysis buffer Solution was sterile filtered after preparation, then aliquoted into 10 mL tubes and stored frozen until use.	
2 x base buffer	1 x
DTT	2 mM (1 mM already from 2 x base buffer)
Triton X-100	2%
sucrose	20%
RNase-free H_2O	(fill up to desired volume)
Sucrose cushion buffer Solution was sterile filtered after preparation, then aliquoted into 10 mL tubes and stored frozen until use.	
2 x base buffer	1 x
Sucrose	30% or 64% (depending on protocol)
RNase-free H_2O	(fill up to desired volume)
Freezing buffer Solution was sterile filtered after preparation, then aliquoted into 2 mL tubes and stored frozen until use.	
2 x TKM buffer	1 x
Roche Protease Inhibitor Cocktail	4 x
PMSF	1 mM
Chloramphenicol	100 µg/mL
Cycloheximide	100 µg/mL
RNase-free H_2O	(fill up to desired volume)
10 x TBE stock solution Solution was sterile filtered or autoclaved after preparation.	
Tris (solid)	0.908 M
Boric acid (solid)	0.89 M
EDTA (stock adjusted to pH 8.0)	20 mM
RNase-free H_2O	(fill up to desired volume)
2 x formamide RNA loading buffer	
Deionized formamide	90%
Tris-HCl (stock adjusted to pH 7.5)	20 mM
EDTA (stock adjusted to pH 8.0)	20 mM
Bromophenol blue	0.04%

Footprint elution buffer Solution was sterile filtered, then aliquoted into 1 mL portions for single-use and incubated at 95°C for 10 min. Stored frozen until use.	
SDS	0.25%
Na-acetate (stock adjusted to pH 5.5)	300 mM
EDTA (stock adjusted to pH 8.0)	1 mM
RNase free H_2O	(fill up to desired volume)
12% denaturing Urea-TBE-PAGE gel buffer Detailed casting procedure is described in section 2.3.1.8.	
Urea (solid)	7.99 M
10 x TBE stock solution	1 x
40% Acrylamide/Bisacrylamide 19:1	12%
RNase free H_2O	(fill up to desired volume)
SYBR Gold staining solution	
SYBR Gold stock solution	1 : 10,000 (v/v)
10 x TBE stock solution	1 x
RNase free H_2O	(fill up to desired volume)
2 x RNaseH hybridization buffer Solution was sterile filtered, then aliquoted into 1 mL portions for single-use and stored frozen until use.	
Tris-HCL (stock adjusted to pH 7.5)	200 mM
NaCl	400 mM
Dynabeads washing buffer	
Tris-HCL (stock adjusted to pH 7.5)	5 mM
NaCl	1 M
EDTA (stock adjusted to pH 8.0)	20 mM
RNase free H_2O	(fill up to desired volume)
Dynabeads RNA wash solution A After preparation, the buffer was supplemented with 0.1% DEPC, incubated overnight at room temperature and autoclaved the next morning. The solution was aliquoted into 5 mL portions and stored frozen until use.	
NaOH	100 mM
NaCl	50 mM
H_2O	(fill up to desired volume)

Dynabeads RNA wash solution B After preparation, the buffer was supplemented with 0.1% DEPC, incubated overnight at room temperature and autoclaved the next morning. The solution was aliquoted into 5 mL portions and stored frozen until use.	
NaCl	100 mM
H_2O	(fill up to desired volume)
20 x SSC buffer After preparation, the buffer was supplemented with 0.1% DEPC, incubated overnight at room temperature and autoclaved the next morning. The solution was aliquoted into 5 mL portions and stored frozen until use.	
Na-Citrate	300 mM
NaCl	3 M
H_2O	(fill up to desired volume)
SSCF buffer	
20 x SSC buffer	1 x
Deionized formamide	20%
RNase-free H_2O	(fill up to desired volume)
8% Non-denaturing TBE-PAGE buffer	
10 x TBE	1 x
40% Acrylamide/Bisacrylamide 19:1	8%
Nuclease-free H_2O	(fill up to desired volume)
SDS-EB lysis buffer The solution was sterile filtered and aliquoted into 950 µL portions in 2 mL tubes. The tubes were incubated at 95° for 10 min denture possible RNase contamination. Aftwerwards, the aliquots were stored at -20°C and immediately after thawing for use, 50 µL proteinase K (20 mg/mL stock) was added per tube.	
Tris-HCl, pH 8.0	50 mM
NaCl	200 mM
EDTA, pH 8.0	20 mM
SDS	2%
Nuclease-free H_2O	(fill up to desired volume)
10 x RNA fragmentation buffer The solution was sterile filtered and stored at -20°C in 100 µL aliquots.	
Tris-acetate, pH 8.3	400 mM
CH_3COOK (potassium acetate)	1 M
$Mg(CH_3COO)_2$ (magnesium acetate)	300 mM
Nuclease-free H_2O	(fill up to desired volume)

2.2 Experimental conditions

2.2.1 *C. reinhardtii* strains used in this work

Throughout this study, the following *C. reinhardtii* strains were used:

Table 2-6: Strains used in this study.

strain	Origin
CC1690	Chlamydomonas Resource Center (Pröschold et al., 2005; Sager, 1955)
A31	(Ramundo et al., 2013)
S11kd2/5	Generated by myself (Gotsmann, 2018, master thesis)

2.2.2 Standard cultivation

C. reinhardtii cultures cultivated under conditions not explicitly explained or described as "standard cultivation" in this work were grown mixotrophically using "TAP" medium adjusted to neutral pH by addition of HCl. TAP medium was formulated according to Table 2-7. Cultivation under standard conditions was facilitated in Erlenmeyer flasks of volume 20 mL up to 2 L always filled up to half of their maximum volume. Flasks were shaken by automatic shakers at 120 rpm under constant illumination at 50 µE and an ambient temperature of 25°C if not indicated otherwise.

Table 2-7: Composition of TAP medium.

Compound	Final concentration	Chemical formula / concentration in stock solution
Tris	20 mM	$H_2NC(CH_2OH)_3$
Beijerinck solution	2.5% v/v	NH_4Cl (299 mM) $MgSO_4$ * 7 H_2O (16 mM) $CaCl_2$ * 2 H_2O (14 mM)
Phosphate buffer	1 mM	KPO_4, pH 7
Trace elements mixture	-	Na_2 * EDTA (25 mM) Zn * EDTA (2.5 mM) $(NH_4)_6Mo_7O_{24}$ (28.5 µM) Na_2SeO_3 (0.1 mM) Mn * EDTA (6 mM) Fe * EDTA (20 mM) Cu * EDTA (2 mM)
Acetic acid	17.5 mM	CH_3COOH

2.2.3 S11 depletion

For depletion of uS11c protein from S11kd2 and S11kd5 cultures, standard cultivation conditions were chosen. To harvest control and knockdown cells from the same culture, cells were grown in 1 L cultures to the logarithmic phase and half of the culture was harvested according to the Ribo-Seq workflow (see section 2.3.1). Afterwards, the remaining cultures were diluted with TAP medium to an OD_{700} of 0.5 and immediately supplemented with 20 µM thiamine-HCL and 20 µg/L vitamin B_{12}. To prevent the cultures from entering the stationary phase, the cells were diluted after 24 h to an OD_{700} of 0.5 and again supplemented with the same concentration of thiamine-HCL and vitamin B_{12} as before. After 48 h of induction, the knockdown samples were harvested as described in the Ribo-Seq workflow.

2.2.4 Phototrophic cultivation in bioreactors

For phototrophic cultivation, cells were grown in Biostream flat-panel benchtop photo-bioreactors in 12 h / 12 h light / dark cycles at 25°C under constant monitoring of medium pH, medium consumption, OD, illumination and aeration. Illumination was calibrated to 100 µE with each 15 min ramping at the beginning and the end of each light phase. The cultures were automatically diluted to maintain an OD that was calibrated to resemble 2 x 10^6 cells / mL.

Table 2-8: Composition of HMP medium.

Compound	Final concentration	Chemical formula / concentration in stock solution
HEPES	5 mM	$C_8H_{18}N_2O_4S$
Beijerinck solution	2.5% v/v	NH_4Cl (299 mM) $MgSO_4$ * 7 H_2O (16 mM) $CaCl_2$ * 2 H_2O (14 mM)
Phosphate buffer	1 mM	KPO_4, pH 7
Trace elements mixture	-	Na_2 * EDTA (25 mM) Zn * EDTA (2.5 mM) $(NH_4)_6Mo_7O_{24}$ (28.5 µM) Na_2SeO_3 (0.1 mM) Mn * EDTA (6 mM) Fe * EDTA (20 mM) Cu * EDTA (2 mM)

To achieve synchronization of the cells, cultures were grown for multiple cycles until a periodic growth pattern in the OD recordings was recognizable, indicating that cell division occurred roughly simultaneously among cells in the culture. The bioreactors were constantly aerated at a rate of 3 L sterile-filtered air per minute.

2.3 Molecular biology workflows

2.3.1 Ribo-Seq workflow

2.3.1.1 Cell harvest

C. reinhardtii cells were cultivated under conditions as indicated for the respective experiments to the mid-log phase (2 – 8 x 10^6 cells/mL). Cells were supplemented with 100 µg/mL chloramphenicol and cycloheximide and immediately poured over silicon ice cubes frozen at -80°C. Cultures were shaken together with the ice cubes until a temperature of 4°C was reached (approximately 2 min). Afterwards, the culture was centrifuged at 4,000 g and 4°C for 2 min and the pellet was washed in 25 mL of TKM+ buffer (Table 2-5) to remove residual medium components. The resuspended cells were centrifuged again with the same settings and the pellet was resuspended in 1:2,000 of the culture's initial volume of freezing buffer (Table 2-5). The suspension was then dripped slowly into liquid nitrogen to yield beads of frozen cell suspension. The beads were stored at -80°C until further processing.

2.3.1.2 Cryo-lysis of cell material

To lyse the cell material, the beads of frozen cell suspension were transferred into liquid nitrogen-cooled containers of a ball mill (Table 2-2) together with 1 cm steel balls. Cells were ground two times with intermittent cooling in liquid nitrogen at 27 Hz for 2 min. The frozen cell powder was stored at -80°C until further processing.

2.3.1.3 Lysate preparation

Frozen cell powders were thawed on ice and mixed with 2 x lysis buffer (Table 2-5) in a 1:1 ratio (weight / volume) and incubated for 5 min at 4°C under constant shaking to dissolve membranes. Subsequently, the lysates were centrifuged at 8,000 g and 4°C for 10 min to pellet cell debris. Supernatants were transferred to fresh tubes either for polysome enrichment (applied to protocols i and ii) or for nuclease digestion directly in the lysate (applied to protocol iii).

2.3.1.4 Polysome enrichment

For enrichment of polysomes, lysates were layered onto ice-cold 2.5 mL sucrose cushion buffer (Table 2-5) of 64% in 8.9 mL polypropylene ultracentrifugation tubes (OptiSeal) and ultracentrifuged for 3 h at 4°C and 60,000 rpm in a Beckman Coulter fixed-angle type 70.1 Ti rotor, corresponding to a centrifugal force of 254,016 g on average. After centrifugation, the supernatants and sucrose cushions were removed carefully und the ribosome pellets were rinsed briefly with 500 µL ice-cold, RNase-free water. Then the pellets were resuspended by incubation in 100 µL ribosome buffer (Table 2-5) overnight on ice followed by thorough pipetting. Insoluble particles were removed from the suspension by 1 min centrifugation at 1,500 g and 4°C and transferring the supernatant to fresh tubes. Samples were either flash-frozen and stored at -80°C until further use or directly processed further.

2.3.1.5 Nuclease digest

For the generation of ribosome protected fragments from polysomes, RNA concentrations of the polysome resuspensions (protocol i and ii) or the lysate (protocol iii) were measured photometrically using a NanoDrop2000 device before adding 1 U of Ambion RNase I and 0.134 U of TURBO DNase were added per µg of total RNA in the respective sample. The samples were incubated for 1 hour each at 4°C (protocol i and iii) or at 23°C (protocol iii). To terminate digestion, 0.4 U of SUPERase•In RNase Inhibitor was added to the samples per unit of previously added nuclease. Afterwards, samples were centrifuged for 10 min at 8,000 g and 4°C and supernatants were transferred to fresh tubes.

2.3.1.6 Monosome purification

Up to 500 µL of lysate or ribosome suspension were loaded on top of 500 µL of 30% sucrose cushion buffer (Table 2-5) in 1 mL open-top thick-wall polypropylene tubes and ultracentrifuged for 30 min at 64,000 rpm in a Beckman Coulter MLA130 rotor or at 72,000 rpm in a Thermo Scientific S150-AT rotor. Subsequently, the pellets were immediately resuspended in 100 µL ice-cold ribosome buffer (Table 2-5) and flash-frozen for storage at -80°C or immediately processed further.

2.3.1.7 Recovery of ribosome protected fragments

The resuspended ribosome pellets were supplemented with 15 mM EDTA pH and thoroughly mixed to achieve dissociation of the subunits of enriched ribosomes and therefore release of the ribosome protected fragments. Immediately, per 100 µL of resuspension, 750 µL of TRIzol

reagent was added to inhibit any potential nucleolytic activity in the suspension by protein denaturation and samples were incubated for 5 min under constant shaking at room temperature. Following, per 100 µL ribosome suspension, 150 µL of pure chloroform were added and samples were thoroughly vortexed and incubated for another 3 min at room temperature. To achieve phase separation of the emulsion, samples were centrifuged for 15 min at 20,000 g at 4°C. The upper phases were transferred to fresh tubes and supplemented with 3 µL of GlycoBlue solution and 0.3 M Na-acetate at pH 5.5 before adding one volume of pure 2-propanol. Samples were briefly mixed by inversion and incubated at -20°C for at least 1 hour to overnight to precipitate nucleic acids from the solution. Next, the precipitates were pelleted by centrifugation for 1 hour at 20,000 g at 4°C. The supernatant was discarded, and the pellet was washed with ice-cold 70% ethanol by brief inversion. After complete removal of the ethanol, the pellet was air-dried for 2 min at 55°C and resuspended in 11 µL of RNase-free water. 1 µL of the solution was measured using the Qubit RNA XR assay according to the manufacturer's protocol for the assessment of purified RNA. The samples were either flash-frozen and stored at -80°C or directly processed further.

2.3.1.8 Casting of urea-TBE gels

12% denaturing Urea-TBE gels were cast by mixing all ingredients of the gel buffer according to Table 2-5 in a 30 mL volume and heating the solution until the urea was dissolved completely without boiling it. Afterwards, 130 µL of 10% APS solution and 8 µL of TEMED solution were added and the solution was immediately poured into 1 mm thick Bio-Rad Protean Mini Gel apparatuses using 10-well combs. After polymerization of the gels, the gel chambers were assembled, and the wells of the gels were thoroughly flushed with 1 x TBE buffer to remove cloaking gel debris.

2.3.1.9 Purification and size-selection of ribosome protected fragments

To purify the ribosome protected fragments from purified bulk RNA, 50 µg of bulk RNA were diluted in 20 µL of RNase-free water and mixed with 20 µL of 2 x formamide RNA loading buffer (Table 2-5). The samples were heated to 80°C for 90 sec to melt RNA structures and immediately transferred to ice. Every sample was loaded in 25 µL portions onto two neighboring lanes of the gel with 5 µL of Dynamarker Prestained for Small RNA Plus as reference. Gels were run at 200 V until the dye front reached the end of the gel. After the run, gels were released from the chambers and glass plates and were incubated in dishes with 5 mL of SYBR-Gold staining solution. Gels were inspected on a blue light table and areas between 23 and 45 nt according to the marker were cut out and transferred to fresh low-bind

tubes. Gel pieces were thoroughly smashed by careful centrifugation through gel breaker tubes (PCR tubes with a hole punched through their tip) and brief freezing of the gel at -80°C. To elute the size-selected RNA from the gel matrix, samples were incubated overnight at 4°C with 400 µL of footprint elution buffer (Table 2-5) upon constant agitation. The next day, the eluted RNA was recovered by separating the gel slurry via centrifugation through Spin-X cellulose acetate columns from the solution according to the manufacturer's manual. The flow-through was transferred to fresh low-bind tubes, mixed with 3 µL of GlycoBlue Coprecipitant and one volume of ice-cold 2-propanol and stored for at least 1 hour to overnight at -20°C to precipitate nucleic acids from the solution. Next, the precipitate was pelleted by centrifugation at 20,000 g at 4°C for one hour and the pellet was briefly washed with 70% ethanol. After the complete removal of ethanol, the pellet was dried for 2 min at 55°C and resuspended in 10 µL of RNase-free water. 1 µL of the solution was measured with Qubit microRNA assay to assess the concentration of RNA. The samples were flash-frozen and stored at -80°C or directly processed further.

2.3.1.10 RNaseH-based rRNA depletion

For rRNA depletion using the RNaseH approach (applied in protocol iii), 2,500 ng of oligo depletion mix (Table 2-4) were mixed with 500 ng of size-selected RNA and diluted with 2 x RNaseH hybridization buffer (Table 2-5) to yield a total volume of 15 µL. The mixture was incubated in a thermocycler at 95°C for 2 min to unfold RNA structures, then the temperature was ramped down to 22°C with a rate of -0.1°C/s and held for 5 min to achieve hybridization of the RNA with the depletion oligos. Next, 2 µL of 10 x RNaseH reaction buffer (Table 2-5) and 3 µL of RNaseH were added to the mix and incubated for 30 min at 37°C to facilitate digestion of RNA-oligo-duplexes. After RNaseH digestion, 3 µL of 10 x TURBO DNase buffer (Table 2-5) and 7 µL TURBO DNase were added and the mixture was incubated another 30 min at 37°C to digest remaining oligos in the reaction. Afterwards, the mixture was purified using the NEB Monarch RNA Cleanup Kit for 10 µg input material according to the manufacturer's protocol. The nucleic acids were eluted from the cleanup column in 36.5 µL RNase-free water and the concentration of RNAs were measured using 1 µL solution in the Qubit microRNA assay (Table 2-3).

2.3.1.11 Streptavidin bead-based rRNA depletion

Per sample, 45 µL of Dynabeads MyOne C1 Streptavidin magnetic beads were washed three times on a magnetic stand by exchanging the buffer and resuspending the beads again in 45 µL Dynabeads washing buffer (Table 2-5). Afterwards, the beads were additionally washed first for 2 min in 45 µL Dynabeads RNA wash solution A (Table 2-5) per sample and then in the same volume of Dynabeads wash solution B (Table 2-5) to inactivate RNases. Finally, the beads were resuspended in 45 µL Dnyabeads RNA wash solution B and for every sample, the

beads were split in two fractions of 30 µL and 15 µL each. For depletion of rRNA fragments, purified RNA was mixed according Table 2-9 to yield a total volume of 20 µL per sample. The mixture then was heated to 80°C in a thermocycler for 5 min to melt RNA structures. Then the temperature was ramped down in 5°C-steps with intermittent holding time of 2 min each to 35°C at a ramping rate of -0.1°C/s to allow hybridization of the rRNA fragments with biotinylated oligos.

Table 2-9: Reaction mixture for rRNA-oligo hybridization.

Component	Final concentration
Ribosome footprints	5 – 10 ng/µL
Deionized formamide	20%
20 x SSC	1 x
EDTA (pH in stock adjusted to 8.0)	0.5 mM
100 µM Biotinylated rRNA depletion mix	25 µM
RNase-free H_2O	Ad 20 µL

After hybridization, the 30 µL portions of prepared beads were magnetized and the buffer was exchanged against the hybridized sample. The samples were incubated for 15 min at ambient temperature with slight agitation to allow binding of rRNA-oligo duplexes to the streptavidin tag of the beads. In the meantime, the 15 µL portions of the prepared beads were magnetized and the buffer was removed shortly before incubation of the sample ends. After incubation, the samples were magnetized, and the supernatant was transferred to the second portion of prepared beads for another 15 min of incubation at room temperature. Next, the samples were magnetized again, and the clear supernatant was transferred to fresh low-bind tubes. Then, each sample was supplemented with 2.5 volumes of ice-cold ethanol and 3 µL GlycoBlue coprecipitant and nucleic acids were precipitated overnight at -20°C. The precipitate was pelleted for 1 h at 20,000 g and 4°C and the pellet was washed with 70% ethanol. After complete removal of the ethanol, the pellet was briefly air dried and resuspended in 41 µL of RNase-free water. Remaining oligos and contaminating DNA fragments were removed by supplementing the sample with 2 µL TURBO DNase, 5 µL 10 x TURBO DNase buffer (Table 2-5) and 2 µL SUPERaseIn RNase inhibitor and incubation at 37°C for 30 min. Afterwards, 150 µL Dynabeads RNA wash solution B and 200 µL phenol-chloroform-isoamylalcohol (PCI) were added and samples were mixed thoroughly. To achieve phase separation of the emulsion, the samples were centrifuged for 20 min at 16,000 g and room temperature. The upper phases were transferred to fresh tubes containing 200 µL chloroform-isoamylalcohol (CI) and samples were mixed and centrifuged again for 10 min at 16,000 g and room temperature. The procedure was repeated a second time before adding 2.5 volumes of ice-cold ethanol and 1 µL GlycoBlue coprecipitant to the supernatant and subsequent precipitation of the RNA at -

20°C overnight. Again, the precipitate was pelleted at 20,000 g at 4°C for 1 h and the pellet was washed twice with 70% ethanol. After complete removal of the ethanol, the pellet was air-dried for 10 min and dissolved in 36.5 µL of RNase-free water. The RNA concentration was determined using 1 µL of the solution in the Qubit RNA HS assay (Table 2-3). The samples were stored at -80°C or directly processed further.

2.3.1.12 5'-phosphorylation of ribosome protected fragments

The purified and rRNA-depleted samples were supplemented with 5 µL 10 x T4-polynucleotide kinase (T4-PNK) buffer, 2.5 µL T4-PNK enzyme and 2 µL SUPERaseIn RNase inhibitor and incubated at 37°C to achieve 3'-dephosphorylation of ribosome protected fragments. After 10 min, the reaction mix was supplemented with 1 mM ATP to start 5'-phosphorylation and the mixture was incubated for an additional 30 min. After incubation, the samples were purified again using the NEB Monarch RNA Cleanup kit (Table 2-3) for 10 µg input according to the manufacturer's protocol. The RNA was eluted in 11.5 µL RNase-free water and the yield was determined using 1 µL via the Qubit microRNA assay (Table 2-3).

2.3.1.13 Ribo-Seq NGS-library preparation and cleanup

NGS-library preparation for Ribo-Seq experiments was carried out using the Perkin Elmer NEXTFlex Small RNA v3 library kit (Table 2-3), strictly following the manufacturer's manual. After preparation, the libraries were further purified via Tris-borate-EDTA polyacrylamide electrophporesis (TBE-PAGE) to remove contaminating adapter-dimer amplification product. The 8% non-denaturing TBE-PAGE gels were casted by mixing 30 mL the gel buffer according to Table 2-5, adding 500 µL of 10% APS and 50 µL TEMED and then pouring the solution into Bio-Rad Protean Mini Gel apparatuses with 10-well combs. After polymerization, the PAGE chambers were prepared and the wells were thoroughly flushed with 1 x TBE running buffer to remove clogging gel debris. Each 25 µL library was mixed with 5 µL TrackIT Cyan/Yellow loading buffer (Table 2-3) and loaded in two 15 µL portions to adjacent wells on the gel. Invitrogen Ultra Low Range DNA Ladder (Table 2-3) was loaded as reference. Gels were run at 200 V until the yellow band of the loading buffer reached the end of the gel. Afterwards, the gels were incubated in SYBR Gold (Table 2-3) staining solution and inspected on a LED light table (Table 2-2). The library bands of approximately 150 bp were cut from the gels and smashed using gel breaker tubes. The libraries were eluted from the gel matrix and cleaned up according to the dedicated section in the manual of the NEXTFlex Small RNA v3 library kit by Perkin Elmer (Table 2-3). After recovery of the cleaned-up libraries, library characteristics

were inspected measuring 1 µL of the solution in the Bioptic Qsep1 Fragment Analyzer (Table 2-2) and library concentration was determined using the Qubit dsDNA HS assay (Table 2-3).

2.3.2 Selective ribosome profiling of cpSRP54

Selective ribosome profiling was performed analogous to the conventional Ribo-Seq protocol with few adaptations. Since DSP was used as crosslinking agent, cells destined for selective ribosome profiling were cultivated in HMP medium, a Tris-free alternative to TAP or TMP media to avoid quenching the reagent (Table 2-8).

2.3.2.1 Preparation of anti-cpSRP54-beads

Prior to coupling beads to cpSRP54 antibody, Amintra protein A resin was washed three times with 100 mM potassium phosphate buffer at pH 7.5 by gently resuspending the beads in the buffer and then pelleting the beads for 1 min at 1,000 g at ambient temperature and subsequent exchange of the supernatant by fresh buffer. To produce “slurry”, approximately 250 µL washed beads were resuspended in 500 µL 100 mM phosphate buffer (Table 2-5) at pH 7.5. To bind the antibody to the resin, 100 µg of purified cpSRP54-antibody was mixed with 100 µL slurry, respectively, and filled up to 1 mL with PBS and 0.1% NP-40 substitute (Table 2-5). The mixture was incubated overnight at 4°C to allow proper binding. Next, the resin was pelleted by 1 min centrifugation at 3,000 g and 4°C and resuspended in 1 mL of PBS supplemented with 0.1% NP-40 substitute by gentle inversion of the tube. The same washing procedure was repeated once more, and the resin was subsequently resuspended in 1 mL of 0.2 M sodium borate buffer at pH 9.0 supplemented with 0.1% NP-40 substitute. Afterwards, the beads were washed three times by pelleting for 1 min at 3,000 g and 4°C and subsequent resuspension in 900 µL of fresh 0.2 M sodium borate buffer at pH 9.0 supplemented with 0.1% NP-40 substitute. The bead-bound antibody was crosslinked to the resin for 30 min at ambient temperature by addition of 22 mM DMP and subsequent constant agitation of the reaction tubes. After crosslinking, the beads were washed twice with 1 mL of 1 M Tris-HCL at pH 7.5 and centrifugation as previously to remove and quench excess DMP. After washing, the antibody-bead conjugate was incubated for 30 min at ambient temperature in 1 M Tris-HCL at pH 7.5 supplemented with 1% BSA to block remaining reactive sites of the resin and quench residual DMP in the solution. Afterwards, the resin was pelleted as before and resuspended in 1 M Tris-HCL at pH 7.5 supplemented with 150 mM NaCl, 1% BSA and 0.01% sodium azide at volumes of 400 µL per 100 µL input slurry used. The resulting aliquots were stored at 4°C until further use. Immediately before usage, the beads were washed twice in TKM+ buffer (Table 2-5) by 15 s centrifugation at 16.000 g and subsequent exchange of the buffer. Finally, the beads were

reconstituted in TKM+ buffer at a final concentration resembling 250 µg input antibody per mL solution.

2.3.2.2 Cultivation

In contrast to the standard protocol, CC-1690 cells for the selective profiling experiment were cultivated under phototrophic conditions in HMP media (Table 2-8) in photo-bioreactors (Table 2-2). Cultures were grown in a 14 h/10 h day/light cycle at 100 µE illumination under constant monitoring of the cultures' optical density and automatic dilution. When both, log-phase growth and synchronization of the cells was indicated by OD-monitoring, the cultures were grown for two more cycles before harvesting.

2.3.2.3 Cell Harvest

To gather large amounts of cell material, for each replicate 2 x 1.5 L of cell culture were sampled from the bioreactor 2 h after illumination phase started. The second sample was only taken after the culture in the bioreactor was filled up with fresh medium and had recovered to log-phase growth and synchronization again. The harvesting procedure was carried out according to the conventional Ribo-Seq protocol (see section 2.3.1.1). After harvesting, the two partial samples for each culture were combined.

2.3.2.4 Sample processing

Cryo-lysis of the samples was carried out according to the conventional Ribo-Seq protocol (see section 2.3.1.2). Afterwards, 4.7 and 3.8 g of cell powder were taken for the replicates A and B and were thawed upon addition of the same amounts of DTT-free, HEPES-based 2 x lysis buffer (Table 2-5) and were incubated with 2 mM DSP for 30 min at 4°C for crosslinking cpSRP54-ribosome complexes. After incubation, DSP was quenched by addition of 100 mM Tris-HCL at pH 8.0. Next, cell debris was removed by centrifugation of the lysate at 8,000 g for 15 min and transferring the supernatant to fresh tubes. After measuring the RNA concentration in via NanoDrop (calibrated against 1 x lysis buffer), the lysate was incubated 15 min at 4°C under constant agitation. Afterwards, 1 U RNaseI per µg of RNA and 0.134 U of TURBO-DNase per µL were added and the mixture was incubated for 1 h at 4°C under constant agitation to facilitate the nucleolytic digestion of polysomes. After incubation, the digest was stopped by addition of 0.4 U of SuperaseIn RNase inhibitor per unit of RNaseI previously added. The antibody-bead conjugate was pelleted by 1 min centrifugation at 1,000 g and 4°C and washed three times with 750 µL TKM+ (Table 2-5) supplemented with 0.1% Tween-20 (Table 2-5) for 2

min at 4°C under constant agitation. Additionally, the three washing steps were supplemented with 10, 5 and 2 µL of SuperaseIn, respectively to avoid overdigestion of RPFs by residual nucleases. Afterwards, the antibody-bead conjugate was reconstituted in 500 µL of TKM+ supplemented with 0.1% of Tween-20 and 2 µL of SuperaseIn and transferred to a fresh tube. The old tube was washed out with the same volume of the buffer to resuspend any residual affinity beads adhering to the tube walls and the solutions were combined. Next, the affinity beads were pelleted by centrifugation at 1,000 g for 1 min at 4°C and washed two times with 750 µL RNase-free TKM+ buffer supplemented with 0.1% of Tween-20 to remove any unspecific binders to the bead-bound antibody. Finally, the beads were pelleted as before and resuspended in 100 µl TKM+ buffer supplemented with 10 mM DTT and 15 mM EDTA to dissociate bead-enriched ribosomes and to dissociate the antibody chains and thus release enriched RPFs into the solution. Immediately, 750 µL of TRIzol reagent were added and the solution was mixed vigorously by vortexing. The solution was incubated for 5 min at room temperature under constant agitation. Afterwards, RNA was extracted from the solution by adding 150 µL chloroform followed by vortexing and 15 min centrifugation at 16.000 g and ambient temperature and then precipitating the upper phase of the emulsion. This was achieved by transferring it to a fresh tube and mixing it with 3 µL GlycoBlue reagent and $\frac{1}{9}$ of the solution's volume of 3 M sodium acetate at pH 5.5 followed by addition of an equal volume of ice-cold isopropanol and incubation overnight at -20°C. The next day, precipitate was pelleted by 1 h centrifugation at 16.000 g and 4°C. The pellet was briefly washed with 1 mL ice-cold 70% ethanol. After removing the ethanol completely, the pellet was dried for 2 min at 55°C with an open lid in a thermo shaker under mild agitation and then dissolved in 11 µL of RNase-free water. The RNA concentration in the solution was determined via Qubit HS RNA assay (Table 2-3) and the samples were directly used for RPF purification analogously to the standard Ribo-Seq protocol (see section 2.3.1.9). The remaining procedure was equivalent to the standard protocol.

2.3.3 RNA-Seq workflow

2.3.3.1 Cell harvest

For RNA-Seq experiments, cells were harvested from mid-log phase cultures cultivated under the specific conditions of the experiment by centrifuging 2.5 to 5 mL of culture for 2 min at 4,000 g. The supernatant was decanted quickly, and the cell pellets were immediately snap-frozen in liquid nitrogen to avoid transcriptomic changes due to the sampling procedure as much as possible. Samples were stored at -80°C until use.

2.3.3.2 Total RNA extraction

To isolate total RNA from frozen cell pellets, the pellets were taken out of the freezer and immediately resuspended in 50°C pre-warmed SDS-EB buffer (Table 2-5) supplemented with proteinase K. The suspension was incubated for 2 min at 50°C under constant agitation and subsequently mixed with 500 µL TRIzol reagent. The mixture was vigorously mixed and incubated for 5 min at ambient temperature under constant agitation. Afterwards, the suspension was transferred to Maxtract High Density tubes and mixed with 200 µL chloroform. The mixture was shaken thoroughly for 15 s and then incubated at ambient temperature for 5 min. The phases of the resulting emulsion were separated by 3 min centrifugation at 16.000 g and ambient temperature. The upper phase was transferred to a fresh tube and mixed with 1.5 volumes of pure ethanol. The solution was used as input to the Macherey Nagel MN NucleoSpin RNA Plant kit at Step 5 of the manufacturers protocol. The remaining procedure was carried out according to the manufacturers protocol and RNA was eluated in 40 µL NRAse-free H_2O. The eluate was stored at -80°C until further processing.

2.3.3.3 RNA fragmentation

The concentrations of total RNA extracts were determined via the Qubit RNA BR assay and quality of the RNA was checked via the Qubit RNA IQ assay. Samples were accepted for further processing, when the IQ value was above 8, indicating a sufficient quality for transcriptomics library preparation. To fragment total RNA, 36 µL of total RNA extracts were mixed with 4 µL of 10 x fragmentation buffer (Table 2-5) and incubated at 85°C for 12 min. Afterwards, samples were immediately transferred to ice and after cooling down, 4 µL of nuclease-free 0.5 M EDTA at pH 8.0 was added.

2.3.3.4 Recovery of fragmented RNA

The fragmented RNA samples were mixed with 360 µL TESS buffer (Table 2-5) and 400 µL PCI. The solution was vortexed vigorously and then centrifuged for 10 min at 16,000 g at 4°C. The supernatant was transferred to a fresh tube and mixed with 1 mL of ice-cold ethanol, 26 µL of RNase-free 5 M NaCl and 2 µL GlycoBlue and then precipitated overnight at -20°C. The next day, the fragmented RNA was pelleted by 1 h centrifugation at 4°C and 16.00 g and the pellet was washed with ice-cold 75% ethanol by gentle inversion of the tube. After complete removal of ethanol, the pellet was dried for 2 min at 55°C with an open lid in a thermo shaker under mild agitation and then dissolved in 20 µL of RNase-free water. The samples were stored at -80°C until use.

2.3.3.5 RNA-Seq library preparation

The preparation of RNA-Seq libraries was carried out using the RiboFree Total RNA library kit by Zymo-Seq (Table 2-3) strictly following the manufacturers protocol. In all cases, 500 ng of input RNA was used. Library concentration and quality were determined subsequently using the Qubit dsDNA HS assay.

2.3.4 Qubit nucleic acid quantification

Nucleic acid quantification of different samples was carried out using the Qubit 4 instrument (Table 2-2) with the following assays, depending on the specific sample type. The following Qubit assays were used, always strictly following the manufacturers protocols: dsDNA HS assay, RNA XR assay, RNA HS assay, RNA BR assay, microRNA assay. In all cases, 1 µL sample input has been used for quantification.

2.3.5 Fragment analyzer NGS-library characterization

NGS libraries were characterized using the Bioptic QSep 1 fragment analyzer (Table 2-2) using the Standard Cartidge (S2) and the 20 bp & 1000 bp Alignment marker (C109100) in the “gDNA (NGS)”-method. Libraries were sequenced only if their electropherogram displayed a suitable length distribution with a peak at ~150 bp, indicating a viable library.

2.3.6 Pooling of sequencing libraries

After measuring library concentrations with the Qubit HS DNA assay and inspecting the library size distribution using the Bioptic QSep 1 fragment analyzer (Table 2-2), selected libraries were pooled together in proportions according to how the sequencing flow cell capacity (400 M reads) was intended to be split among the samples. In all cases, specific caution was applied to select only libraries, whose sequencing barcodes were compatible with each other in a library pool.

2.3.7 Next Generation Sequencing

In all cases, NGS-libraries were sequenced on an Illumina NextSeq 550 system (Table 2-2) using the Illumina NextSeq 500/550 v2.5 reagent kit (Table 2-3) with a capacity of 400 million clusters. For RNA-Seq libraries, a read depth of 20 million reads was targeted, while for Ribo-Seq libraries, a read depth of 40 million reads was targeted. All samples were sequenced with

75 cycles in single-end mode and demultiplexing of the sequencer output was performed using bcl2fastq v2.20.0.422 (Table 2-1).

2.3.8 NGS data analysis

Analysis of NGS bulk data was carried out in a Linux environment on a server with 24 Intel XEON SP 6126 CPUs and 384 GB RAM. The analysis of NGS data was carried out in two steps: First, raw sequencing data in FASTQ format were trimmed, filtered and converted to genome-aligned reads in SAM/BAM format. Second, aligned reads were analyzed transcript-wise by taking the genomic annotation into account to yield expression values, ribosome profiles and corresponding quality control parameters. Both steps were carried out using the Anaconda platform utilizing multiple environments for the specific steps of the process.

2.3.8.1 Raw NGS data processing

Raw data processing started with raw FASTQ files obtained from demultiplexing reads after the sequencing run. Reads from the FASTQ files were first trimmed using Cutadapt (Martin, 2011) to remove the 3'-adapters. Afterwards, UMI-tools (Smith et al., 2017) was used to remove the unique molecular identifiers from both ends of the read. Following, Cutadapt was used again to filter out all reads shorter than 20 nt or longer than 39 nt to remove reads that were unlikely to represent true RPFs. The fact that both, 3'- and 5'-adapters of the NEXTFLEX Small RNA Sequencing kit contained a 4-nt UMI lead to the situation that all reads must contain a UMI at their 5'-end and additionally at their 3'-end, if the length of the read after trimming was 73 nt or shorter, indicating that the complete read insert had been sequenced. The double-usage of UMI-tools resulted in erroneous read headers that were repaired by a custom python script (see section 8.5). Next, the reads were mapped against a set of non-coding RNA-sequences using STAR to remove ncRNAs from the data sets. Genomic information of ncRNAs was retrieved from Ensembl Plants (Dyer et al., 2025). After ncRMA removal, the remaining reads were mapped against the *C. reinhardtii* genome v6.1 using STAR again. Resulting BAM files were deduplicated with UMI-tools and indexed with Samtools to enable further processing. The complete procedure was carried out in a bash script (see section 8.4 for the complete script including all commands and all parameters used).

2.3.8.2 Ribo-Seq functions

If not stated differently, all downstream analyses of BAM files were carried out using Ribo-Seq functions (RSf) interactively in a Jupyter Notebook. A detailed documentation of the tool set

and explanations of its functions is given in section 2.3.8.4 and the complete source code is given in section 8.3.

2.3.8.3 Differential expression analysis

(Seabold & Perktold, 2010)(Benjamini & Hochberg, 1995)Differential expression analysis was carried out using the R-package DESeq2 (Love et al., 2014) and R-Studio as programming environment. As input for DESeq2, read count tables from RSf representing only main transcripts were used. Following differential expression analysis, all calculated p-values were corrected using the Benjamini-Hochberg method (Benjamini & Hochberg, 1995) with an FDR of 0.05. To correct for bias in the data, a LOWESS curve was calculated for each data set using the averaged log2-fold expression of control and treatment for each transcript as x and the log2-fold change as y (Cleveland, 1981). The resulting curve was used to normalize the amplitude of change for each transcript when linear regression of the curve resulted in a coefficient of determination lower than 0.99 or if either the slope or the x-offset of the regression line was higher than 0.01, indicating that the LOWESS curve did not sufficiently resemble a flat line along the x-axis. For calculations, the Python package statsmodels was used (Seabold & Perktold, 2010). Afterwards, all transcripts with an absolute log2-fold change of 1 and an FDR lower than 0.05 were considered as significantly changed.

2.3.8.4 Ribo-Seq functions documentation

The following paragraph contains a short documentation of RSf, briefly describing how to call the functions, their parameters and working principles. To use the tools, the riboseq_functions.py file has to be copied into the working directory of the programming environment used and to be imported into the script or notebook. For the sake of simplicity, it is recommended to import RSf using the alias “rs” as shown below:

```
import riboseq_functions as rs
```

Dependencies: Linux OS, python 3.8.13, pysam 0.16.0.1, pandas 1.4.3, numpy 1.20.3, matplotlib 3.3.4, seaborn 0.11.2, Python standard library

Building the genomic database:

Function

```
rs.prepare_genomic_coordinates(path_to_annotation, delimiter, skiprows=0,
comment="#",      parenttag="Parent",      idtag="ID",      nametag="Name",
genetag="geneName", transcripttag="mRNA", organism="Chlamydomonas")
```

path_to_annotation: Here specify the path to the annotation file of the genome in GFF3 format.

delimiter: Here specify the delimiter that is used in the annotation file to separate values. Usually tabulator “\t” or comma “,”.

skiprows: If the annotation file contains any lines in the header section that are not part of the annotation itself, specify the number of these lines here. Default 0.

comment: If the annotation file contains any commenting lines, make sure these lines are marked with a distinct symbol at the beginning and specify this symbol here. Default “#”.

parenttag: Here specify the keyword that indicates the ID of a Parent entry in the annotation file. Default “Parent=”.

idtag: Here specify the keyword that indicates the ID of an entry in the annotation file. Default “ID=”.

nametag: Here specify the keyword that indicates the trivial name of a transcript in the annotation file. Default “Name=”.

genetag: Here specify the keyword that indicates the trivial name of a gene in the annotation file. Default “geneName=”.

transcripttag: Here specify the keyword that indicates the category of an entry as transcript in the annotation file. Default “mRNA”.

organism: Applies function logic specifically refined to the *C. reinhardtii* genome v6.1 (“Chlamydomonas”) or to the human genome hg38 (“human”). If a different genome is used, stay with “Chlamydomonas”.

Returns: A pandas.DataFrame object containing condensed information about every transcript annotated in the specified annotation file. The result is automatically saved as a byte string object in the pickle format (“.pkl”) in the directory that contains the original GFF3 file. Also directly returns the pandas.DataFrame object.

This function reads in a genomic annotation in GFF3 format using the pandas module. Taking advantage of the parent - child relationships between untranslated regions / coding sequences, exons, transcripts and genes that are used in GFF3 annotation, the function extracts all annotated transcripts and for each transcript individually extracts all information from the annotation file. The result is a tabular genomic database in form of a pandas.DataFrame object representing every annotated transcript with all exons, UTRs, coding sequences etc. in one row. The table is automatically saved in the pickle format, ensuring the database remains always unaltered, unless intended. For later use, the database can simply be loaded using the

“pandas.read_pickle()” command. The function also directly returns the pandas.DataFrame object, allowing to save it directly into a variable for direct use.

Instantiate the Transcript objects for the whole genome:

Function

rs.BuildTranscripts(genomic_coordinates, samfile, chlamy=True)

genomic_coordinates: Here specify a variable containing the genomic database in form of a pandas.DataFrame object or “pandas.read_pickle(path-to-pickle-file)”.

samfile: Here specify a variable containing the BAM/SAM file of interest in form of a pysam.AlignmentFile object or “pysam.AlignmentFile(path-to-BAM/SAM-file)”.

chlamy: Set to “False” if not working with *C. reinhardtii*. Triggers the execution of extra functions refined for *C. reinhardtii*.

Returns: A Python dictionary containing all annotated transcript IDs as keys and the respective Transcript objects as values. Prints out the number of objects created, and the time passed during the process.

This function browses the genomic database and for every row instantiates a Transcript object that is linked to the specified SAM/BAM file and stores them in a Python dictionary with their genomic ID as key. At this point, the objects only contain the genomic coordinates of the transcripts they represent and the total number of reads within the SAM/BAM file they belong to. This is done on purpose to allow the user to work with selected transcripts only without having to do time-consuming calculations always on the whole data set. If working with *C. reinhardtii*, chlamy=True triggers the assembly of the mature psaA transcript from its three autonomously annotated exons and calculates its coverage in coding sequence and full transcript mode. For these processes, two extra objects “psaA_full” and “psaA_cds” are added to the dictionary. Furthermore, chlamy=True triggers the calculation of full transcript and coding sequence coverage of both psbA copies and then executes the extra function “combine_psbA_copies”, which combines the coverage of both gene copies within their identical coding sequence region and appropriately recalculates the coverage of both copies in full transcript and coding sequence mode.

The Transcript class:

Class

constructor:

```
rs.Transcript(chromosome, strand, five_utr, three_utr, codings, exons,
name, identifier, samfile, info, sort=True)
```

Returns: An rs.Transcript object representing a specific transcript containing all positional information about this transcript and the respective info line from the annotation file. The object is linked to the SAM/BAM file of interest.

The rs.Transcript object is the central part of RSf, providing multiple methods to calculate different transcript-specific metrics that are tailored especially for questions in context of the Ribo-Seq method. These metrics and the parameters they are based on are stored in each object as attributes. At the time of instantiation, each object only has a limited set of attributes defined which are continuously complemented with new attributes using an object's methods. Table 2-10 lists all attributes of the rs.Transcript class and their meaning.

Table 2-10: List of all rs.Transcript object attributes and the methods that create them.

attribute	Python data type	description
Standard set of rs.Transcript attributes created upon instantiation.		
.chromosome	string	genomic identifier of a chromosome
.strand	string	coding strand "+" or "-", for psaA "x"
.five_utr	list of tuples of two integers	list of exon coordinates encoding a 5'-UTR
.three_utr	list of tuples of two integers	list of exon coordinates encoding a 3'-UTR
.codings	list of tuples of two integers	list of exon coordinates encoding a CDS
.exons	list of tuples of two integers	list of all exon coordinates of a transcript
.name	string	trvial name of a transcript, if existent. Otherwise genomic identifier
.identifier	string	genomic identifier of a transcript
.sort	boolean	"True" or "False". Indicates wether to sort genomic coordinates of the transcript upon instantiation of the object
.samfile	pysam.AlignmentFile object	object containing the reads and mapping information of a BAM / SAM file
.info	string	info line of a transcript as given in the annotation
.totreads	integer	number of reads within the data set the rs.Transcript object is linked to
.mreads	float	.totreads divided by 1,000,000
Extended set of rs.Transcript attributes created by class methods.		

attribute	Python data type	description
Attributes created by the rs.Transcript.fulltranscript() / rs.Transcript.full_psaA() method.		
.full_exonborders	list of integers	contains the last position of each exon relative to the transcripts first position
.full_featureborders	list of integers	contains the last position of 5'-UTR, CDS and 3'-UTR relative to the transcripts first position
.full_coverage	dictionary	contains each position of the transcript and the number of reads mapping to this position as key / value pair
.full_readnames	list	contains the header of all reads mapping to the transcript
.full_feature_readlist	list of three lists	contains lists of all reads mapping to 5'-UTR, CDS and 3'-UTR of the transcript separately
.full_poscounter	integer	counting value used during calculations
.full_gbc	list of ten floats	contains the transcripts extent of gene body coverage for thresholds 1 to 10
.full_length	integer	the full length of the mature transcript in nucleotides
.full_reads	integer	the number of reads mapping to the transcript
.full_rpkm	float	RPKM value of the transcript
.full_cpm	float	CPM value of the transcript
.full_feature_readnums	list of three integers	contains the numbers of reads mapping to 5'-UTR, CDS and 3'-UTR of the transcript
Attributes created by the rs.Transcript.only_cds() / rs.Transcript.psaA_cds() method.		
.cds_exonborders	list of integers	contains the last position of each exon relative to the CDS's first position
.cds_coverage	dictionary	contains each position of the CDS and the number of reads mapping to this position as key / value pair
.cds_readnames	list	contains the header of all reads mapping to the CDS
.cds_poscounter	integer	counting value used during calculations
.cds_gbc	list of ten floats	contains the CDS's extent of gene body coverage for thresholds 1 to 10
.cds_length	integer	the full length of the mature CDS in nucleotides
.cds_reads	integer	the number of reads mapping to the CDS
.cds_rpkm	float	RPKM value of the CDS
.cds_cpm	float	CPM value of the CDS
Attributes created by the rs.export_profile() method.		
.profile	two- or three-dimensional list	contains lists of lists that feature- and exon-wise store positions and their coverage for plotting
Attributes created by the rs.psite_profile() method.		
.p_table	pandas.DataFrame object	contains P-site counts versus positions of a CDS and basic statistics
Attributes created by the rs.periodicity() method.		

attribute	Python data type	description
.firstperiodicity	pandas.DataFrame object	contains P-site counts versus positions and some additional information for the first x codons
.lastperiodicity	pandas.DataFrame object	contains P-site counts versus positions and some additional information for the last x codons
.firsttuples	list of tuples of two integers	contains the exon and region borders used to define the CDS positions to take into consideration for calculation of the first x codons' periodicity
.lasttuples	list of tuples of two integers	contains the exon and region borders used to define the CDS positions to take into consideration for calculation of the last x codons' periodicity
.periodicity_table	pandas.DataFrame object	contains P-site counts versus positions and some additional information for the user-defined number of first and last codons of the CDS

Each of the attributes listed in Table 2-10 can be accessed via rs.Transcript.[attribute name], enabling the user to develop customized analysis workflows on his own.

Methods of the Transcript class:

Class method

rs.Transcript.only_cds(calc_gbc=False, get_readnames=False)

calc_gbc: Set to "True" to calculate the extent of gene body coverage for cut-off values 1 to 10 for a transcript. Default "False".

get_readnames: Set to "True" to save the identifiers of the reads mapping to the coding sequence of the transcript of interest. Default "False".

Returns: Calculates and sets the attributes cds_readnames, cds_gbc, cds_length, cds_reads, cds_rpkm, cds_cpm and returns them.

This class method extracts all reads that map to the coding sequence of the transcript of interest to calculate RPKM, CPM, read number and length of the respective ORF. Upon request, the method also calculates the percentage extent of gene body coverage of that ORF for cut-off values from 1 to 10 and saves the identifiers of the mapping reads to a list.

Class method

rs.Transcript.fulltranscript(calc_gbc=False, get_readnames=False)

calc_gbc: Set to “True” to calculate the extent of gene body coverage for cut-off values 1 to 10 for a transcript. Default “False”.

get_readnames: Set to “True” to save the identifiers of the reads mapping to the full transcript of interest. Default “False”.

Returns: Calculates and sets the attributes full_readnames, full_gbc, full_length, full _reads, full _rpkm, full _cpm and returns them.

This class method is the equivalent to rs.Trascript.only_cds() for calculations considering the whole transcript instead of only the coding sequence.

Class method

rs.Transcript.export_profile(mode=”cds”)

mode: Must be “cds” if a ribosome profile of the coding sequence alone is requested, otherwise “full” to generate a profile containing all features of the transcript. Default “cds”.

Returns: A tuple containing a (multidimensional) list of feature- and / or exon-wise coverage and positional data relative to the transcript or CDS start (depending on mode) and the mode used for calculation. Stores the result in the object as attribute “profile”.

This class method can be run only after rs.Transcript.fulltranscript() or rs.Trascript.only_cds() has been called on the respective Transcript object. “full” mode can only be called if rs.Transcript.fulltranscript() has been called before, “cds” mode works with both. The method sorts the calculated coverage information by exon and genomic feature and stores the result as “profile” attribute in the object which later can be accessed by the rs.plot_coverage() function for proper display.

Class method

rs.Transcript.psite_profile(fiveoffsets)

fiveoffsets: Must be a Python dictionary containing all read lengths occurring in the data set as keys and their respective 5’-P-site offsets as values.

Returns: A pandas.DataFrame object containing the position, codon number, summed number of P-sites mapping to each codon, each frames preference per codon, average / median read length for each position and respective standard

deviation and frame of each position of the transcripts coding sequence. Also sets the result as “p_table” attribute.

This class method is used to produce P-site profiles of full open reading frames. It is necessary to calculate 5’-P-site offsets first using the rs.offset_calc() function and to create a Python dictionary containing the selected offsets as values to their respective read length species as keys.

Class method

rs.Transcript.periodicity(fiveoffset, profilesize=(30,30))

fiveoffsets: Must be a Python dictionary containing all read lengths occurring in the data set as keys and their respective 5’-P-site offsets as values.

profilesize: Here, specify a tuple containing the number of codons in the front and in the rear of the transcript’s ORF that should be considered for metagene P-site calculation.

Returns: Does not return a value. Instead calculates the P-site profile for the first i and last n positions of it the transcripts ORF as indicated in the profilesize parameter and sets the result as attribute “periodicity_table”, containing (relative) position, P-site counts, (relative) codon number, sum of P-sites per codon and frame preference per codon in form of a pandas.DataFrame object . If the ORF is too short for the given profilesize parameter, the frame preference, sum of P-sites per codon and P-site counts will simply be set to 0.

This class method works very similar like rs.Transcript.psite_profile(), but acts only on limited regions in the front and the rear of the ORF. The method is intended to be called on selections of transcripts, for example the top 1000 expressed transcripts, in a loop to sum up the individual periodicity_table attributes of all selected transcripts to build a metagene P-site profile.

Class method

rs.Transcript.full_psaA()

Returns: Coverage, read identifiers, transcript length, gene body coverage, read number, RPKM and CPM for the psaA mature transcript and sets the respective attributes.

This class method is intended to be called only by the rs.BuildTranscripts() function when working with the *C. reinhardtii* genome. The method assembles the mature full-length psaA transcript from its trans-spliced exons and calculates proper expression values etc.

Class method

rs.Transcript.psaA_cds()

Returns: Coverage, read identifiers, transcript length, gene body coverage, read number, RPKM and CPM for the psaA mature coding sequence and sets the respective attributes.

The same as rs.Transcript.full_psaA(), but limited to the coding sequence of the mature transcript.

Function

rs.export_basic_information(
transcript_objects, mode="cds", gene_body_coverage=False)

transcript objects: Must be a Python dictionary containing rs.Transcript objects to be analyzed.

mode: Here, specify if the data set is to be analyzed in "full" or in "cds" mode. Default "cds".

gene_body_coverage: Set to "True" if information calculating gene body coverage is desired.

Returns: A pandas.DataFrame object reresenting every rs.Transcript object analyzed in one row containing identifier, trivial name, chromosome, strand, transcript / ORF length in nucleotides, reads mapped to the transcript / ORF, RPKM, CPM and the respective info line from the genomic database. If gene_body_coverage is set to true, 10 additional columns are returned containing the extent of gene body coverage for cut off values 1 to 10.

This function is intended to do quick and basic analyses of whole data sets by calling the "rs.Transcript.fulltranscript()" or "rs.Transcript.only_cds()" method on every Transcript object in the data set and returning the most basic information in tabular format that can be exported directly as spreadsheet. While the "full" mode is intended for RNA-Seq analysis, for Ribo-Seq data the "cds" mode should always be used.

Function

```
rs.FPlength_dist(
sample, genomic_coordinates, readlengths=(20,39), organelles=["plastome",
"mitogenome", "chrM", "mito", "CP", "Cp", "Mt", "MT", "chloroplast",
"mitochondrium", "plastid", "mitochondria", "plast"])
```

sample: Must be a path to the BAM / SAM file to be analyzed.

genomic_coordinates: Here, specify a variable containing the genomic database in form of a pandas.DataFrame object or "pandas.read_pickle(path-to-pickle-file)".

readlengths: Here, specify the maximum and minimum read lengths to be considered in a tuple in format (min, max). Default (20,39).

organelles: Here, specify a Python list of chromosomal identifiers to be considered organellar chromosomes. Default =["plastome", "mitogenome", "chrM", "mito", "CP", "Cp", "Mt", "MT", "chloroplast", "mitochondrium", "plastid", "mitochondria", "plast"].

Returns: A list containing two pandas.DataFrame objects, containing all considered read length species and their respective frequencies as raw counts (index 0) or percentage values (index 1) for each chromosome separately and for all detected organelles separately.

This function is called on a BAM / SAM file to calculate the distribution of read length species in a sample. The resulting list contains two pandas.DataFrame objects, of which the first contains the raw counts for all considered read length species as rows and columns for each chromosome (including mitochondrial and / or chloroplast chromosome) plus one column summing up all chromosomes that are not indicated as organellar thus representing cytosolic reads. The second object provides the same information normalized for each column by its total count. The "organelles" parameter has a default list of common identifiers for mitochondrial and plastid genomes. In case the organellar identifiers in the genome used are not among this selection, the identifiers have to be given in form of a Python list to the "organelles" parameter.

Function

```
rs.biotype_length_dist(
sample, genomic_coordinates, readlengths=(20,39), organelles=["plastome",
"mitogenome", "chrM", "mito", "CP", "Cp", "Mt", "MT", "chloroplast",
"mitochondrium", "plastid", "mitochondria", "plast"], threshold=0.5,
chromosomes=False, binsize=10000, fast=True, center=True)
```

sample: Must be a path to the BAM / SAM file to be analyzed.

genomic_coordinates: Here, specify a variable containing the genomic database in form of a pandas.DataFrame object or "pandas.read_pickle(path-to-pickle-file)".

readlengths: Here, specify the maximum and minimum read lengths to be considered in a tuple in format (min, max). Default (20,39).

organelles: Here, specify a Python list of chromosomal identifiers to be considered organellar chromosomes. Default =[“plastome”, “mitogenome”, “chrM”, “mito”, “CP”, “Cp”, “Mt”, “MT”, “chloroplast”, “mitochondrium”, “plastid”, “mitochondria”, “plast”].

threshold: Here, specify the minimum share a specific biotype must have on a read’s alignment to be counted as mapping to this biotype if the “fast” parameter is set to “False”. Default 0.5.

chromosomes: Here, specify the chromosomes to analyze by their identifiers in a Python list. If set to “False”, all chromosomes are analyzed. Default “False”.

binsize: Performance factor. Can be any integer number >1. Generally should be in the range between 10,000 and 100,000. Default 10,000.

fast Indicate if fast calculation is desired at the expense of correctly categorizing borderline reads, can be “True” or “False”. Default “True”.

center: Indicate if read centering should be applied in the case of fast calculation, can be “True” or “False”. Default “True”.

Returns: A list containing four pandas.DataFrame objects, containing all considered read length species and their respective frequencies as raw counts for each chromosome separately and for all detected organelles separately. Index 0 object contains only reads categorized as coding region mapping, index 1 object contains only five prime untranslated region mapping reads, index 2 object contains only three prime untranslated region mapping reads and index 3 object contains all remaining reads, categorized as intergenic region mapping.

This function is called on a BAM / SAM file to calculate the distribution of read length species among the different “biotypes” or genomic features (coding sequence, five prime untranslated region, three prime untranslated region, intergenic region) for all or specified chromosomes and organelles. This operation is default run in “fast” mode. In this case, with “center” parameter set to “True”, every read’s center position (in case of even read length: $\frac{\text{read length}}{2} + 1$) is determined and the read is categorized as the respective biotype this position is mapped to. In case the “center” parameter is set to “False”, every read’s left-most mapping position is taken instead (in case of + encoded reads this is the 5’-most position, in case of – encoded

reads this is the 3'-most position). In case the "fast" parameter is set to "False", for every read the portion mapping to any biotype is calculated and the read is categorized as any biotype its mapping-to portion is exceeding the value of the "threshold" parameter. Of note, a read can fall into multiple categories if it is mapping to a genomic position that encodes different biotypes on different transcript variants or if the "threshold" variable is lower than 0.5, for example allowing a read to be categorized both as coding sequence and five prime untranslated region, if it partially covers both of these biotypes. Vice versa, a read can also be completely ignored if the "threshold" variable is greater than 0.5. This only applies if the "fast" parameter is set to "False". Otherwise, a read will always be categorized only by the mapping location of its center position or its left-most mapping position. If this position maps to two different biotypes in different transcript variants, the categorization always favors coding sequence categorization over untranslated regions and these in turn over intergenic regions. Generally, fast mode with centering should be accurate enough for most analyses while threshold mode might be useful for specific questions and filtering. Since this operation is computationally very expensive, the calculation is carried out in chunks with the "binsize" parameter defining the size of these chunks as number of base pairs of the genome that are analyzed in each chunk. This strategy greatly improves RAM consumption and processing time. 10,000 proofed to be a well-balanced value for this parameter for the *C. reinhardtii* genome, but may vary across different genomes, if the chromosome sizes differ greatly. If only specific chromosomes are of interest for analysis, the user can specify these chromosomes with a Python list containing the respective chromosomal identifiers to avoid having to take the whole genome into consideration.

Function

```
rs.offset_calc(
dataset,    five_offset=True,    three_offset=True,    readlengths=(20,39),
min_five=20,            min_three=20,            chloroplast="plastome",
mitochondria="mitogenome")
```

dataset: Must be a Python dictionary containing rs.Transcript objects to be analyzed.

five_offset: Here, specify if calculation of the 5'-P-site offset is desired. Default "True".

three_offset: Here, specify if calculation of the 3'-A-site offset is desired. Default "True".

readlengths: Here, specify the maximum and minimum read lengths to be considered in a tuple in format (min, max). Default (20,39).

min_five: Here, specify the minimum number of bases the five prime untranslated region of a transcript should have to be considered in the calculation. Default 20.

min_three: Here, specify the minimum number of bases the three prime untranslated region of a transcript should have to be considered in the calculation. Default 20.

chloroplast: Here, specify the chromosomal identifier or the chloroplast genome, if existent. Default: “plastome”.

mitochondria: Here, specify the chromosomal identifier or the mitochondrial genome, if existent. Default: “mitogenome”.

Returns: A list containing six pandas.DataFrame objects, representing count tables for read offsets in the following order: chloroplast 5’-P-site offset, mitochondrial 5’-P-site offsets, cytosolic 5’-P-site offsets, chloroplast 3’-A-site offsets, mitochondrial 3’-A-site offsets, cytosolic 3’-A-site offsets.

This function is called on a Python dictionary containing rs.Transcript objects to extract the reads covering start codons and five prime untranslated regions or the three prime untranslated regions to record the length of their 5’-P-site or 3’-A-site offsets. The calculation is done separately for every organelle due to the specification of the chromosomal identifiers of the chloroplast and mitochondrial genome. For every organelle and every offset type, a count table is returned containing the frequencies of every offset length observed for any read length species. By analyzing these tables, the user has to decide by himself which offset is considered the true offset of a read length species. While this usually is the offset with the largest frequency, this rule does not always hold true and must be decided from case to case. If the user still wants to apply this rule, an offset dictionary can be constructed very easily using the following code:

```
offset_tables = rs.offset_calc(dataset)
cytosolic_five_table = offset_tables[2]
cytosolic_five_offsets = {}
for col in cytosolic_five_table.columns:
      cytosolic_five_offsets[col] = cytosolic_five_table[col].idxmax()
```

Function

```
rs.plot_coverage(
transcript,    save=False,    savepath="",    show=True,    scale=False,
scaleto=10000000)
```

transcript: Must be an rs.Transcript object.

save: Here, specify if the resulting ribosome profile should be saved to disk as image, can be “True” or “False”. Default “False”.

savepath: Here, specify the directory intended for saving the image, if “save” is set to “True”.

show: Here, specify if the ribosome profile should be displayed directly in the programming environment. Can be “True” or “False”. Default “True”.

scale: Here, specify if the ribosome profile should be scaled to a specific data set read depth, can be “True” or “False”. Default “False”.

scaleto: Here, specify the desired data set read depth the ribosome profile should be scaled to, if “scale” is set to “True”.

Returns: A matplotlib.Figure object representing the ribosome profile of the analyzed rs.Transcript object. If indicated, the function also saves the image to disk in a format indicated by the file extension of the savepath (e.g. name.pdf, .jpg etc.). If no file extension is indicated, the image is saved in bitmap format.

This function plots the coverage along a coding sequence or a complete transcript indicating the untranslated regions and different exons of the coding sequence in different colors. In case of Ribo-Seq data is the ribosome profile. It can only be called on a rs.Transcript object after the rs.Transcript.fulltranscript() / rs.Transcript.only_cds() and the rs.Transcript.export_profile() methods have been called on the same object before to construct the profile. It is important to note that rs.plot_coverage() only interprets the output of these methods but does not calculate the profiles itself. According to the specific parameters set, the function can either directly display the ribosome profile or save it to disk. If the “save” parameter is set to “True”, the full path of the desired directory should be specified in the “savepath” parameter. Otherwise, the image will be saved in the current working directory. A full workflow to saving a ribosome profile to disk and displaying it in the programming environment could look like the following:

```
import pandas
import pysam
import riboseq_functions as rs

x = pandas.read_pickle(path_to_pickled_genomic_database)
y = pysam.AlignmentFile(path_to_bam_file)
data_set = rs.BuildTranscripts(genomic_coordinates=x, samfile=y)
t = "identifier_of_transcript_of_interest"
data_set[t].fulltranscript()
data_set[t].export_profile(mode="full")
rs.plot_coverage(data_set[t], save=True, savepath="home/transcript.pfd")
```

Function

```
rs.multiplot(
identifier, path, datasets, fmt="pdf")
```

identifier: Must be a string representing the genomic identifier of the transcript of interest.

path: Here, specify if the path of the directory where to save the resulting image.

datasets: Must be a list of Python dictionaries that contain rs.Transcript objects representing the transcript that is supposed to be plotted.

data_set_names: Here, specify a Python list containing the names of the data sets to analyze in the same order as given in the list passed to the "datasets" parameter. Default "undefined".

fmt: Here, specify the desired format of the image by its file extension. Default "pdf".

Returns: Saves an image of normalized coverage profiles across the coding sequence of the same transcript in multiple different data sets plotted together as line plots.

This function is a legacy function to rs.multiplot2(). Generally, rs.multiplot2() provides more options and is recommended to use rather than rs.multiplot(). However, since this function could still be useful in some cases, it was kept in the module. rs.multiplot() simply plots the coverage profile of the same transcript in multiple different data sets as lineplot. To achieve comparability between profiles of data sets with different read depth, the coverage per nucleotide is normalized by summed coverage of all nucleotides for each profile separately. The resulting image is saved to the directory indicated in the "path" parameter in a specified format using the trivial name of the transcript of interest. If no trivial name is available for a transcript, the genomic identifier is used instead. If the name of the data sets are not specified in the "data_set_names" parameter, the legend on the final image will display the data sets' indexes within the list given to the "datasets" parameter instead. Unlike the rs.plot_coverage() function, rs.multiplot() will autonomously call the rs.Transcript.only_cds() method on the Transcript objects of interest before creating the plots if this has not been done previously by the user.

Function

```
rs.multiplot2(
identifier, datasets, path=False, names=False, label="A-B",
mode="difference", on="cds")
```

identifier: Must be a string representing the genomic identifier of the transcript of interest.

datasets: Must be a list of Python dictionaries that contain rs.Transcript objects representing the transcript that is supposed to be plotted.

data_set_names: Here, specify a Python list containing the names of the data sets to analyze in the same order as given in the list passed to the "datasets" parameter. Default "undefined".

fmt: Here, specify the desired format of the image by its file extension. Default "pdf".

Returns: Saves an image of normalized coverage profiles across the coding sequence of the same transcript in multiple different data sets plotted together as line plots.

Function

```
rs.compare_profiles(
identifier, controls, treated, normalization="scale", scaleto=10000000,
on="cds")
```

identifier: Must be a string representing the genomic identifier of the transcript of interest.

controls: Must be a list of one or more Python dictionaries that contain a rs.Transcript object representing the transcript that is supposed to be analyzed. These data sets are used as the control group.

treated: Must be a list of one or more Python dictionaries that contain a rs.Transcript object representing the transcript that is supposed to be analyzed. These data sets are used as the experimental group.

normalization: Here specify the type of normalization that should be applied to the data. Must be "scale" or "internal". Default "scale".

scaleto: Here specify the desired total data set read depth to scale the data to. Default 10,000,000.

on: Here specify if the whole transcript or the coding sequence only should be considered for analysis. Must be "cds" or "full". Default "cds".

Returns: A Python list containing two pandas.DataFrame objects containing the normalized coverage data and respective mean and standard deviation per nucleotide for the control (index 0) and experimental group (index 1) separately.

This function is intended to compare the coverage profile of the same transcript in two different experimental groups containing multiple replicates each. The function will normalize each data sets coverage profile either by artificially scaling it to the total data set read depth indicated in the “scaleto” parameter or by normalizing every nucleotide’s coverage to the total sum of all position’s coverage in that profile. Afterwards, the average coverage and standard deviation of every nucleotide position is calculated for both experimental groups separately. The results can be plotted using external packages to compare the ribosome profiles under different conditions and to identify changes in the translation dynamics.

Function

```
rs.fasta_to_dict(
fasta, headersymbol=”>”)
```

fasta: Must be a string representing the path to a fasta file that contains a genomic sequence.

headersymbol: Must be a string representing the symbol that characterizes the header line of a chromosome in the fasta file. Default “>”.

Returns: A Python dictionary containing the chromosomal identifiers as keys and their respective sequence as values.

This function is intended for convenient conversion of a FASTA encoded genome sequence into a Python dictionary. In this way, the genomic sequence can be directly accessed by the user though the chromosomal identifier and positional information for indexing the sequence. After running the function, it is recommended to save the dictionary to disk using the pickle module. This allows to reopen the genome without to recompile it every time. The Python dictionary returned by this function is needed as input to the “grab” function to extract sequence information from the genome.

Function

rs.grab(
genome, identifier, dataset, spec=False, start=0, stop=0, feature="coding")

genome: Must be a Python dictionary representing every chromosomes sequence as value to their genomic identifier as key.

identifier: Must be a string representing the genomic identifier of a transcript whose sequence is supposed to be extracted.

dataset: Must be a Python dictionary containing the rs.Transcript object that represents the transcript to be analyzed.

spec: Here specify if the extraction of a specific sequence part is desired. Can be "True" or "False". Default "False".

start: If "spec" parameter is set to True, specify the first position of the extracted sequence to return (relative to the start position of the transcript's or feature's sequence). Default 0.

stop: If "spec" parameter is set to True, specify the last position of the extracted sequence to return (relative to the start position of the transcript's or feature's sequence). Default 0.

feature: Must be a string indicating the feature of the transcript whose sequence is suppes to be extracted. Can be "coding", "all", "five_utr" or "three_utr". Default "coding".

Returns: A Python list containing in the following order: the coding sequence, amino acid sequence, RNA-sequence, original genomic sequence, of a transcript.

This function is intended to extract sequence information about a specific transcript of interest directly from the genomic sequence. In case a coding sequence or a full transcript sequence was queried, the function returns the respective amino acids sequence as well as the original genomic sequence, the coding sequence alone and the RNA-sequence (containing Us instead of Ts). The function always considers splicing and +/- orientation of the transcript.

Function

rs.peakfinder(
peaklist)

peaklist: Must be a Python list containing the positions in a coverage profile that surpasses a user defined threshold in ascending order.

Returns: A Python list containing tuples with start and end positions of presumable peaks in a coverage profile.

This function is a very primitive algorithm to identify peaks using nothing more than a list of index values. This list has to be generated by the user according to his definitions. The function browses the list for consecutively rising index values. If a streak of at least two consecutively increasing values has been found, the first and the last of these values are returned within a tuple in the final list. A meaningful input to this function could be the list of positions in a ribosome profile, where the coverage exceeds the median coverage of the whole profile. Whenever a relevant peak occurs in this profile, a streak of consecutively increasing positions will surpass this threshold and will be found by the algorithm. This function is intended to aid automated peak detection in ribosome profiles or difference profiles to enhance the user's capabilities to detect changes in translation dynamics across the whole genome of an organism.

3 Results

3.1 Establishment of Ribo-Seq and exploration of the *C. reinhardtii* translatome

3.1.1 Development of a toolset for interactive tailored Ribo-Seq data analysis

A problem for the downstream analysis of many high throughput methods and especially Next Generation Sequencing (NGS) techniques are the limited capabilities to interact with the data. The usual way to analyze NGS data is by using established bioinformatic command line tools which may output gene expression data or lists of differentially expressed genes. However, only few of these tools are developed to analyze Ribo-Seq data and are adapted to the special challenges posed by this technique. As an example, for transcriptomics analysis the distribution of reads across a transcript may be interesting only in very special cases. However, in Ribo-Seq data such distributions reflect translation dynamics and therefore are of utmost interest. Although a variety of tools exist to specifically analyze Ribo-Seq data, many of them are not suited to distinguish RPFs of different organelles, provide limited portability across different organisms and file types or simply are unusable due to outdated dependencies or neglected maintenance. Also, functionalities of these tools can differ a lot, necessitating the use of multiple tools in a row for specific analyses. On top of these problems, most available Ribo-Seq tools are command line tools or scripts of static nature. If so, they usually follow a rigid analysis workflow that cannot be adjusted by the user to the specific needs of an experiment.

Due to these reasons, I developed a custom set of functions and classes useful for Ribo-Seq analysis using the Python programming language. This collection called “Ribo-Seq functions” (short: RSf) allows for flexible and dynamic analyses of Ribo-Seq data sets and is most efficiently used interactively in Jupyter Notebooks for custom exploratory data analysis. Moreover, RSf can also be used to create static Python scripts for standardized workflows when a general analysis path for a data set has been worked out. Apart from a basic analysis workflow for transcriptome wide calculation of expression values and gene body coverage, RSf provides multiple functions to analyze specific transcripts of interest in depth.

RSf heavily relies on the Python package pysam, a Python wrapper of the HTSlib C-API that allows to interact with SAM and BAM files in a very fast manner. Moreover, RSf makes extensive use of the Python modules Pandas and NumPy for efficient calculations and data manipulation. The core of RSf is the “Transcript” class, a class that is instantiated for a

transcript of interest and contains all information that is available about this transcript in the genomic annotation of the organism. Most importantly, the Transcript object stores the positional information of the transcript's locus including the chromosome, strand and coordinates of exons, untranslated regions and coding sequences. In this way, the user can access a BAM file and extract all reads that map to the respective genomic coordinates and distinguish the genomic features they map to in splice-aware manner. Based on the extracted alignments, RSf calculates read counts, expression values, ribosome profiles and P-site profiles and stores them as attributes in the Transcript object. RSf allows this in "only_cds" mode for accurate expression values of Ribo-Seq data, considering only coding sequence coordinates or in "fulltranscript" mode also covering the untranslated regions to analyze full transcripts or conventional RNA-Seq data. Moreover, for the trans-spliced psaA transcript in the *C. reinhardtii* chloroplast, RSf assembles a mature, full-length transcript to calculate meaningful values for the whole transcript rather than for each of the three exons separately. Although this function is specifically adapted to the *C. reinhardtii* chloroplast genome, similar functions for other trans-spliced genes can be created analogously.

Transcript objects are instantiated for the whole organism's genome through querying a table that contains all information about all annotated coding transcripts of that organism. This table must be built only once by reading in the organism's genomic annotation in GFF3 format and can then be saved for further use. Apart from the Transcript object, RSf contains functions to analyze a data set on the metagene level by analyzing multiple Transcript objects at once in respect to RPF length distribution, RPF biotype distribution, P-site offset calculations and metagene P-site profiles. For comparison of multiple data sets at once, RSf provides the utility to scale, average and compare ribosome profiles of the same transcript between multiple data sets as well as plotting them together. If detailed analysis of a specific transcript is desired, the user is aided with the ability to read in the genomic sequence of the organism and a function that extracts specified loci and returns the respective genomic, transcriptomic and the corresponding amino acid sequence also considering splice junctions. Finally, the tool set is completed by a simple peak finder algorithm for the automated analysis of ribosome profiles and a function to plot ribosome profiles of single transcripts with specific exon-wise and feature-wise coloring. To provide maximum flexibility to the user, the functions in all cases return Python objects that can be stored in variables for further processing and calculations, and all attributes of the Transcript object are accessible individually after definition, allowing the user to develop his own customized functions, as needed. It is highly recommended to run this module on a high-performance server that provides large capacities of memory. While simple analyses of single BAM files during this work required only small amounts of RAM (~10 GB), the simultaneous in-depth analysis of four BAM files was observed to consume up to 120 GB. RAM consumption in general depends on the size of the organism's genome in question,

especially on the number of annotated coding transcripts. Due to the specific biological interest of this work, RSf contains many specific adaptations to the *C. reinhardtii* genome v6.1, but in general can be used for any organism, if a genomic annotation in GFF3 format exists. The full tool set consists of about 2,000 lines of pure Python code only and is appended to the supplementary material of this work.

3.1.2 Technical comparison of three nuclease digests

One major aim of this work was the establishment of a Ribo-Seq workflow optimized for deep analyses on the sub-codon level. To achieve this goal, three slightly different protocols (referred to as protocols or samples i, ii and iii hereafter) were applied to samples of *C. reinhardtii* wild-type strain CC-1690 grown under mixotrophic conditions (continuous illumination at 50 µE, acetate rich TAP medium, Table 2-7) and compared with each other to find the most suitable protocol. Protocols i and ii were based on a variety of classical published protocols (Becker et al., 2013; B. Y. Chung et al., 2015; Hsu et al., 2016; Ingolia et al., 2009; Oh et al., 2011) with polysome purification prior to a 1 h nuclease digest with 1 U recombinant RNase I per µg of applied RNA at 4°C (sample i) or 23°C (sample ii). Sample iii was prepared according to an adapted protocol with nuclease digest (1 h digest with 1 U/µg RNA at 4°C) directly in the cell lysate and subsequent monosome purification. This protocol was intended to be further augmented with an immunoprecipitation protocol for factor selective ribosome profiling if successful (see section 3.3). Due to the complicated nature of Ribo-Seq workflows, exceptional care must be taken to assess the quality of such data sets. This must be done in terms of technical quality (e.g., sequencing depth, ncRNA contamination, number of unmapped reads) to control technical problems in the Ribo-Seq workflow but also in terms of biological quality. The biological quality parameters read length-distribution, biotype-distribution and degree of tri-nucleotide periodicity are especially useful in this context and help to estimate if a data set is suitable to answer questions that require deep analysis. Comparing the three nuclease digests in terms of technical quality, the loss of reads within every step of the raw read processing pipeline was determined and plotted (Figure 3-1 A). The input read number into the pipeline was comparable for all samples (i.: 52,485,208 reads, ii.: 53,906,634 reads, iii.: 49,082,591 reads). Of the three samples, sample iii with 19.56% yielded the greatest proportion of unique reads. This value was slightly smaller in sample ii but reduced almost by half in sample i. Interestingly, while for samples ii and iii ncRNA reads represent the greatest loss of sequencing depth (ii: 46.62%, iii: 51.57%), in sample i, filtering the reads for RPF-typical length range (20 to 39 nt) is responsible for the largest losses (54.42%). Because of this imbalance, the proportion of reads excluded due to ncRNA mapping is considerably smaller in

i (30.98%). However, analyzing the proportion of filtered reads in relation to the number of input reads at every processing step separately (Figure 3-1 D), it becomes evident that the degree of ncRNA contamination among reads in the RPF-typical length range is comparable to the other samples and even slightly greater. Examining the portions of reads that were filtered out in the length filtering step (Figure 3-1 C), the great loss of read depth in sample i during length

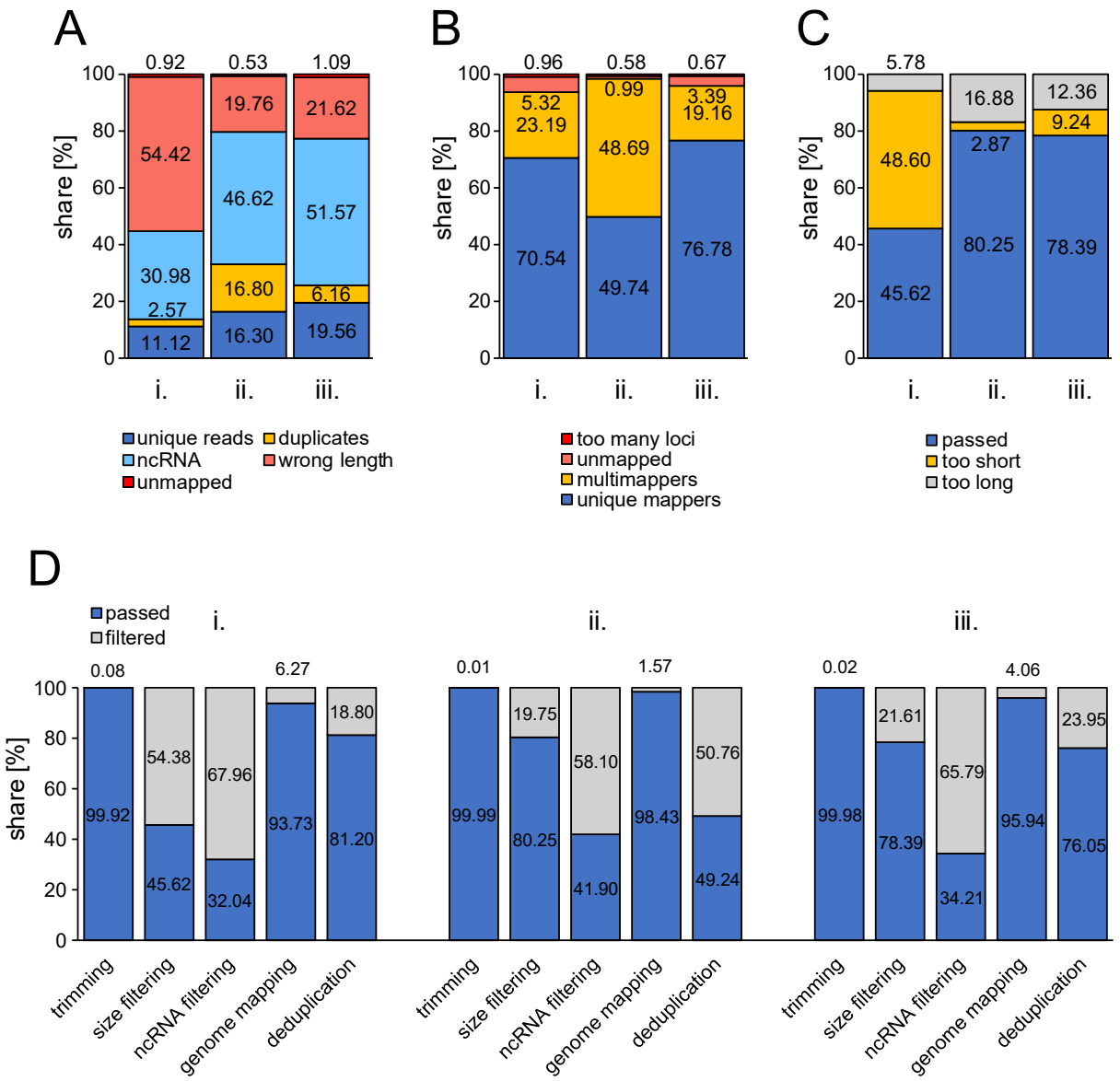


Figure 3-1: Technical quality assessment of Ribo-Seq data sets. (A) Stacked column chart representing the share of Ribo-Seq reads in each data set excluded during the read processing pipeline due to specific reasons (yellow, cyan, light red and deep red) and the amount of unique reads left over at the end of the pipeline (blue) in relation to the respective pipeline read input. Pipeline input: i.: 52,485,208 ii.: 53,906,634 iii.: 49,082,591. (B) Stacked column chart representing basic mapping statistics of the three data sets as share of the total read number that was passed to the genome mapping step for each data set separately with mapped read portions shown in blue and yellow and excluded read portions shown in red shades. (C) Stacked column chart representing the portions of reads that passed length filtering (blue) or were too long (grey) or too short (yellow) to be considered true RPFs in relation to the respective number of reads that was passed to the length filtering step. (D) Column chart representing the portion of reads for each data set and processing step that were filtered out (grey) or passed on to the next processing step (blue) in relation to the respective number of reads that was input to the respective step. Columns are ordered after the sequence of processing steps in the pipeline.

filtering can be attributed to a substantial portion of reads that were excluded due to a length smaller than 20 nt. The portion of too short reads in sample i is multiple times larger than in the other two samples. Given that the nuclease digest of sample ii was performed under much harsher conditions that should favor smaller reads, this observation may seem counter intuitive. However, considering that the sequencing libraries are prepared from RNA fragments that are length-separated via gel electrophoresis and excised from urea-TBE gels (Table 2-5), it is likely that variations in the running behavior of these gels or in the excision of the fragments are responsible for these effects rather than the nuclease digest itself.

Although samples ii and iii are similar regarding the loss of reads during length filtering and ncRNA removal (Figures 3-1 A and 3-1 C), the relative amount of multimapping reads is more than two times higher in sample ii compared to sample iii (Figure 3-1 B). Multimapping reads are expected to occur in Ribo-Seq data sets, especially due to the short length of RPFs (~30 nt) and are randomly distributed among the equally suited mapping positions in this pipeline. However, the difference between sample ii and the other two might point to a true technical bias in this data set. An explanation could be that the harsh nuclease digest may yield shorter RNA fragments which therefore have a higher probability of mapping equally well to multiple distinct positions in the genome. In fact, the average mapping length for the genome mapping step was slightly lower for sample ii (28.49 nt) than for the others (i: 29.88 nt, iii: 29.52 nt) (Supplemental Table 1). Moreover, the relative content of duplicates is also more than two times higher in sample ii compared to both others (Figure 3-1 D). Duplicates are the consequence of suboptimal amplification during library preparation. Since deduplication is performed after mapping, duplicates could also represent a second factor inflating the portion of multimapping reads if these are overrepresented among the duplicates. The number of genomic loci found to be covered by duplicate reads in sample ii is slightly lower compared to sample iii (Supplemental Table 1), suggesting that the additional PCR duplicates do not arise from a general increase of over-amplification in sample ii compared to sample iii. Even though this is no direct indicator that the increase of multimapping reads is connected to the increase in duplicates, it implies that the increase in PCR duplicates may be attributable to over-amplification of specific genomic regions. The read losses during the trimming and the genome mapping steps were comparable in all three samples and were in a low range, indicating that none of the digest conditions poses a risk to sequence or mapping quality in general. At the end of the read processing pipeline the data sets contained 5,834,757 (i), 8,784,507 (ii) and 9,602,968 (iii) mapped unique read alignments.

Cytosol, chloroplast, and mitochondria all possess their own types of ribosomes (80S for cytosol, 70S for chloroplast and 55S for mitochondria), that have evolved their own specific shapes and compositions. Due to this heterogeneity among the different organelle's

ribosomes, it is to be expected that their RPFs reflect these differences in their read length distributions. Nevertheless, it is also to be expected that these distributions should range narrowly around a specific central RPF length that reflects the proportions of each respective ribosome species. It is important to note that the nuclease digest can have serious impact on the RPF length and thus must be carefully balanced to avoid under- or over digested RPFs as far as possible since these negatively affect the calculation of 5'-P-site- and 3'-A-site-offsets. Displayed in Figure 3-2 A, the RPF length distributions differed considerably between the three conditions and organelles. The RPFs originating from cytosolic ribosomes displayed one clear peak in all three conditions with a maximum at 31 nt (i), 28 nt (ii) and 30 nt (iii) length, respectively. The chloroplast derived RPFs in contrast showed three broad peaks with maxima at 24-25 nt, 28 nt and 33-35 nt that appeared differently clear depending on the condition. With two distinct peaks with maxima at 21-22 nt and 33-35 nt and a third peak only present in sample ii with a maximum at 27 nt, the mitochondrial RPFs also showed a quite different length distribution. At this point of analysis, it was not clear why the organellar RPF distributions differed so much from the cytosolic RPF distribution. However, the RPF quality regarding length distribution was chosen to be assessed by the distribution of the cytosolic RPFs, since these made up more than 95% of RPFs in each data set (see Supplementary Table 2). While sample iii's cytosolic RPF distribution formed a sharp peak ranging from 27 to 33 nt with a maximum at 30 nt, the distributions of the other samples were less sharp with their peaks at 31 nt (i) and 28 nt (ii). The peak of sample ii's distribution showed a prominent shoulder at 30 nt, indicating that this might be the true main RPF length and that its distribution with 28 nt maximum might be the result of over digestion. Due to the harsh digest conditions applied to the sample (1 h digest at 23°C), this explanation seems plausible. Sample i also showed a rather broad distribution with a maximum at 31 nt, which in turn would suggest slight under digestion (1 h digest at 4°C). Sample iii instead displayed a very sharp peak with a maximum at 30 nt, that very likely represents the true RPF length for *C. reinhardtii* cytosolic RPFs. Given the assumption that sample i is under digested, this observation is remarkable in respect to the fact that the concentration of RNase I in sample i was much higher in terms of units per volume than in sample iii. This is because 1 U RNase I was applied per µg of RNA in all samples and the volume of sample i after ultracentrifugation and resuspension of the polysomal pellet was considerably smaller than the volume of the lysate used in condition iii. The exact reason why condition iii (1 h digest in lysate at 4°C) yields the sharpest RPF distribution remains elusive, but it can be speculated that the concentration of polysomes by ultracentrifugation might cause polysomes to form aggregates which are less vulnerable to the nuclease digest than free floating polysomes in a cell lysate. Regarding sample ii, this would mean that the elevated temperature during digestion might have been sufficient to outweigh such an effect and cause an over digest, nonetheless. Since sample iii displayed the most

promising RPF length distribution, further specific analyses on RPF lengths were restricted to this sample.

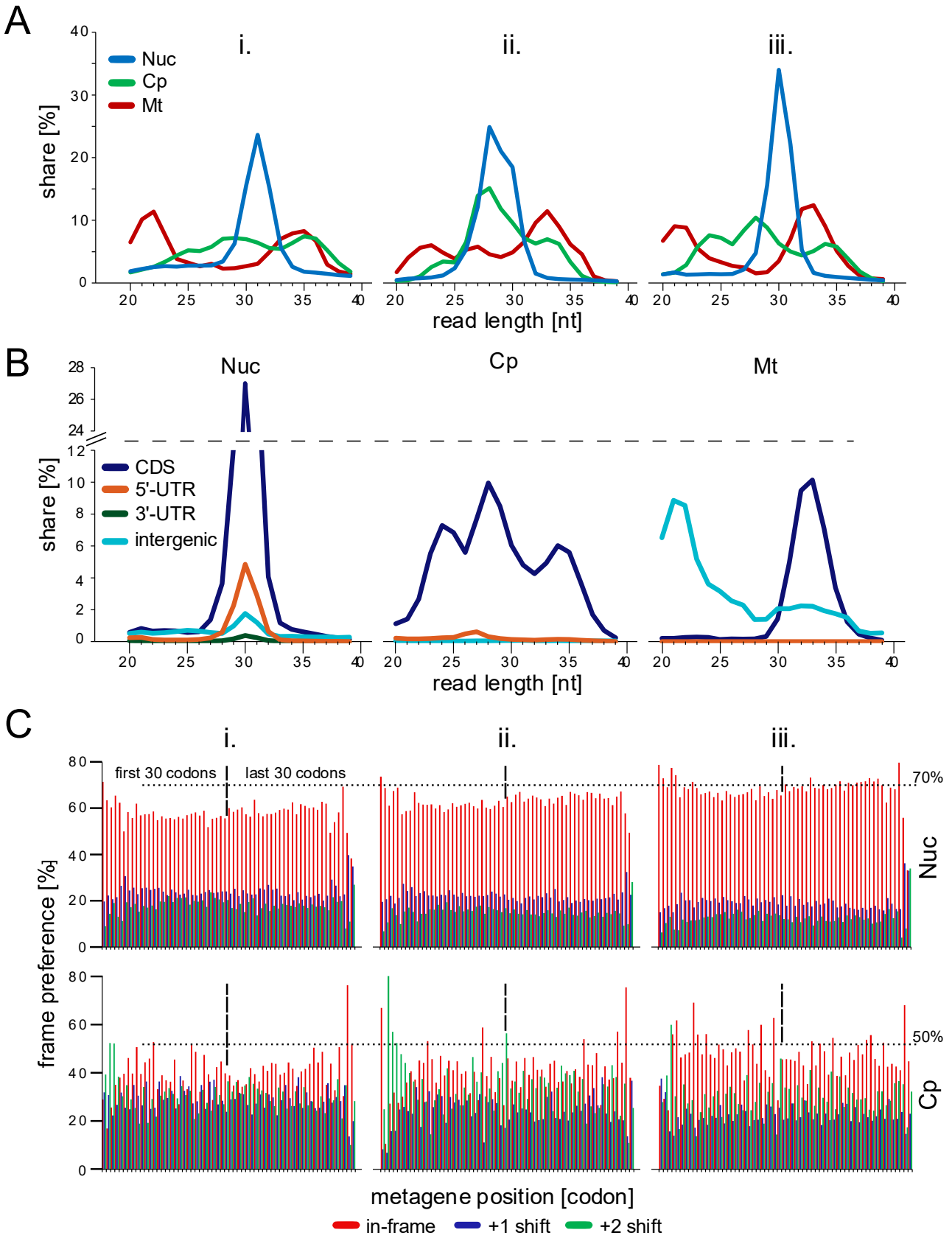


Figure 3-2: Biological quality control parameters. (A) Line graphs representing the distribution of RPF length species in all three data sets for each organelle separately. Values represent the percentage share of the respective organelle's RPFs. (B) Line graphs representing the distribution of RPFs mapping to genomic positions of different biotypes per organelle in sample iii. Values represent the percentage share of the respective organelles RPFs. (C) Metagene P-site frame preference of nuclear (Nuc, upper panel) and chloroplast (Cp, lower panel) encoded transcript's RPFs within the first 30 and last 30 codons of the 2000 top-expressed transcripts in all three samples. For each codon, the red bar represents the portion of P-sites mapping in-frame to this codon, while blue and green bars represent P-sites shifted by one or two nucleotides relative to their translated ORF, respectively. Modified from Gotsmann et al., 2024.

Apart from the pure RPF length distribution, it is also important to monitor the specific biotype of the genomic regions the RPFs map to. Since the known noncoding RNAs were removed during the read processing pipeline, the remaining possible biotypes were classified as 5'-untranslated region, 3'-untranslated region, coding sequence and intergenic region. Since ribosomes should mainly bind to coding sequences of transcripts, it is to be expected that the vast majority of RPFs map to this biotype. If an RPF spanned two biotypes, as frequently observed for ribosomes during translation initiation (5'-untranslated region and coding sequence), the RPF was categorized as the biotype that was covered predominantly. In case an RPF covered two biotypes in 50/50 proportions, the categorization followed the hierarchical order of coding sequence, 5'/3'-untranslated region, intergenic region.

Figure 3-2 B displays the length distributions for cytosolic, plastid and mitochondrial RPFs in sample iii separately per biotype as portion of the total amount of RPFs for each organelle. Coding sequences (CDS) indeed proved to be the predominant biotype among cytosolic and plastid RPFs, but among mitochondrial RPFs instead a considerable amount was identified as intergenic, especially in the length range of 20 to 30 nt. Since their length distribution differs dramatically from the coding sequence mapping RPFs, they likely originate from unannotated noncoding RNAs the data set was not filtered for during the read processing pipeline. Interestingly, the mitochondrial RPFs that were mapped to coding sequences formed a sharp peak after all with a maximum at 33 nt, as expected for true RPFs. Among the cytosolic RPFs, 5'-untranslated regions were the most prominent contaminating biotype, followed by intergenic RPFs. Since their length distribution also ranges around 30 nt, it is to be expected that these really represent true RPFs originating from the cytosolic 5'-scanning mechanism upon translation initiation, misannotated coding sequences (5'-UTR) or from unannotated open reading frames (intergenic). The chloroplast RPFs instead barely showed any contaminating biotype. Considering that both cytosolic and mitochondrial RPFs showed a clear sharp length distribution, it is very unlikely that the digest failed to produce true RPFs in the chloroplast. If the trimodal RPF length distribution reflects ribosome shapes in the chloroplast, this would suggest a high degree of ribosome heterogeneity in the organelle. The very few RPFs derived from 5'-UTRs range around a length of 27 nt, possibly reflecting the dimensions of the chloroplast translation initiation complex bound to the start codon. Of note, RPFs mapping to 3'-untranslated regions were virtually absent in all three organelles. This is to be expected of high quality Ribo-Seq data, since translation is usually terminated when a stop codon enters the ribosomal A-site, causing dissociation of the ribosome. Due to this mechanism, rarely more than 50% of an RPF's sequence should map to a 3'-untranslated region. The very few RPFs found mapping to 3'-untranslated regions might be the consequence of erroneous stop codon read through, a phenomenon that takes place due to codon misreading (Y. Zhang et al., 2024)

or errors in the genomic sequence that may mistakenly lead to the annotation of stop codons within open reading frames.

To interpret Ribo-Seq data on codon level and below, it is necessary to determine the frame preference of the RPFs. Since the translocation step during elongation is faster than the decoding and peptidyl transferase reaction, it is expected that most ribosomes should have a codon bound in-frame with the decoded ORF at their P-site. To determine the presumable position of the ribosome's P-site on an RPF, the 5'-P-site or 3'-A-site offsets (the distance between the 5'- or 3'-end of an RPF and the 5'/3'-terminal position of the P-site) are determined. This is done in a simple approach by sorting all RPFs that fully enclose either a start or a stop codon by their length and determining the offset between the first base of the start / stop codon and the 5'/3'-end for each of these RPFs individually. After that, the most frequently occurring offset for each RPF length species is assumed to be the true distance between this RPF length species' end and the first position of its P-site. This approach especially profits from the occurrence of accumulations of RPFs (called "initiation peaks" hereafter) that specifically cover start codons, usually yielding an exceptionally high coverage in these areas, thus leading to prediction of 5'-offsets with high accuracy. However, in the chloroplast, these initiation peaks were generally much less pronounced than in the cytosol, leading to dubious 5'-offset predictions. This problem was solved by estimating the 3'-offset instead as it is frequently done for other species (Lauria et al., 2018). For mitochondria, offset calculation failed since no 5'-untranslated regions are annotated for the mitochondrial chromosome in the current *C. reinhardtii* genome annotation (v6.1), massively aggravating 5'-P-site offset calculation. Moreover, too few RPFs mapped to the annotated 3'-untranslated regions to confidently calculate 3'-A-site offsets. Since the mitochondrial genome of *C. reinhardtii* only encodes eight protein-coding transcripts, it was therefore neglected in this analysis. After offset determination, the P-site positions of all RPFs mapping to the 2000 top-expressed transcripts of the nucleus and all chloroplast transcripts were determined and for each RPF, the frame of the codon found at the P-site relative to the respective ORF was determined. Since ribosomes are unable to translocate into 5'-direction during elongation, these frames were defined as shifted forward by one or two nucleotides if not in-frame. The counts for each frame were summed up for each codon across all considered transcripts for the first and the last 30 codons of the respective ORFs and normalized by the sum of all frames' counts per codon individually to yield the codon-wise metagene frame preference (Figure 3-2 C) of chloroplast and cytosolic RPFs under the three conditions. For cytosolic RPFs, the in-frame preference was very high in all samples with sample iii having the best ranging around 70% (average: 68% in-frame, 19% +1 shift, 13% +2 shift). For chloroplast RPFs, the frame preference was rather unclear in sample i, but sample ii and especially sample iii displayed an acceptable preference for in-frame RPFs with sample iii again having the highest values

ranging around 50% (average: 47% in-frame, 23% +1 shift, 30% +2 shift). Of note, if the heterogeneity of chloroplast RPF length species may be attributable to ribosome heterogeneity, which might negatively influence P-site site determination due to the mixing of RPFs of different ribosome specific origin in one RPF length category, thus skewing the offset calculation. For most codons, in-frame RPFs strongly outnumbered the other frames, leaving a codon-wise reoccurring pattern on the plots displayed in Figure 3-2 C, known as “tri-

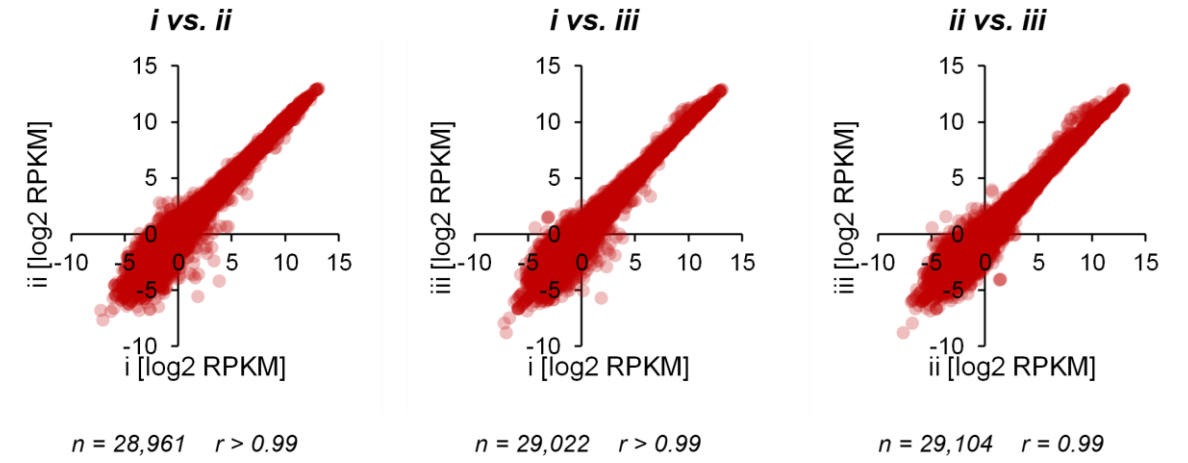


Figure 3-3: Correlation of expression values between samples. Scatter plots opposing the distribution of log2-RPKM values (number of reads per transcript, normalized by the transcript’s length in kilobases) in all three samples against each other. n indicates the number of transcripts taken into consideration (all transcripts with non-zero RPKM value in both opposing samples) and r indicates Pearson’s correlation coefficient between two samples.

nucleotide periodicity” in the field (Ingolia et al., 2009). This phenomenon is a desirable marker for Ribo-Seq data of highest quality and suggests the data is viable for interpretations even on the sub-codon level.

Protocol iii consistently surpassed the other two approaches in all aspects of bioinformatical quality control. In terms of technical quality control, sample iii did not exhibit a specific desirable trait but combined a reasonable number of unique reads after deduplication with an acceptable portion of reads in the RPF-typic length range and of reads mapping to multiple locations on the genome. On the other hand, sample iii had the largest share of contaminating ncRNA in respect to its unfiltered total read depth. However, since all these qualities are influenced by technical variation during the wet lab workflow, none of these can be attributed to the nuclease digest alone. Due to sample iii’s excellent biological features and a considerably faster and easier workflow, as well as the possibility to use it as basis for factor selective ribosome profiling, protocol iii was further on used as the standard workflow for Ribo-Seq experiments.

Although the three conditions yielded data sets of notably different qualities, they proved to be surprisingly comparable on the level of expression values (Figure 3-3). All three samples showed a very high Pearson’s correlation coefficient of 0.99 or greater comparing each other,

indicating a strong monotonic linear relationship between the samples' expression values. This indicates that even though the three digest conditions yielded data sets of varying RPF qualities, these variations do not negatively influence the determination of genome wide expression values to quantify the translatome. However, if an experiment is intended to answer questions of translation dynamics on the codon level of a transcript, it is recommended to aim for highest RPF quality, especially to enable accurate P-site calculation.

Often, the depth of RNA-Seq data is simply described by the number of genes or transcripts that were covered by a certain minimum number of reads in a data set. Since the information of RNA-Seq experiments is usually reduced to a simple expression value per transcript, this is a reasonable approach. However, in a Ribo-Seq data set the distribution of reads across an ORF reflects the translational dynamics of this ORF. Due to this, Ribo-Seq data provides much more valuable information that needs to be taken into consideration when the quality of a data set is assessed. It can be assumed that information content of a Ribo-Seq data set is proportional to the number of transcripts quantified and the completeness of coverage across their ORFs. Therefore, to analyze the depth of data sets used in this study, the extent of the so-called "gene body coverage" for every transcript was determined. This percentage value simply represents the portion of an ORF that is covered by at least a certain number of reads and can be seen as approximation for information content available about the specific transcript's translational dynamics. Categorizing all transcripts into groups from 0% to >90% gene body coverage in 10% steps allows to assess how well transcripts are covered. Gradually increasing the read number that is used as the coverage threshold in this context also allows to estimate the basal coverage of transcripts in the data set by tracking how the numbers of transcripts within the distinct categories (0% to >90%) change with increasing threshold. The general idea about this approach is that the extent of gene body coverage is higher, the fewer gaps are found within their coverage and the extent of gene body coverage is more stable against the increasing cut off, the higher its basal coverage is. As depicted in Figure 3-4 A, almost half of all annotated transcripts (15,824 transcripts, including alternative variants) were covered to an extent of more than 50% with at least one read in sample iii, indicating a good depth in general. Even after increasing the coverage cut-off to 10 reads, still more than half of all annotated transcripts had detectable coverage, indicating that the basal coverage within the data set was also good. With a cut-off of 10 reads 1,508 transcripts could still be identified as covered to an extent of more than 90%, mainly representing transcripts with high translational activity (heat maps for samples i and ii in Supplementary Figure 1). Although this group certainly contains many of the transcripts with the highest expression values, these alone are not sufficient to identify the most completely covered transcripts. In Figure 3-4 B, visualizing the log2-transformed RPKM values versus their extent of gene body coverage at cut-off 10, a subtle tendency for higher RPKM values towards a higher gene body coverage can be

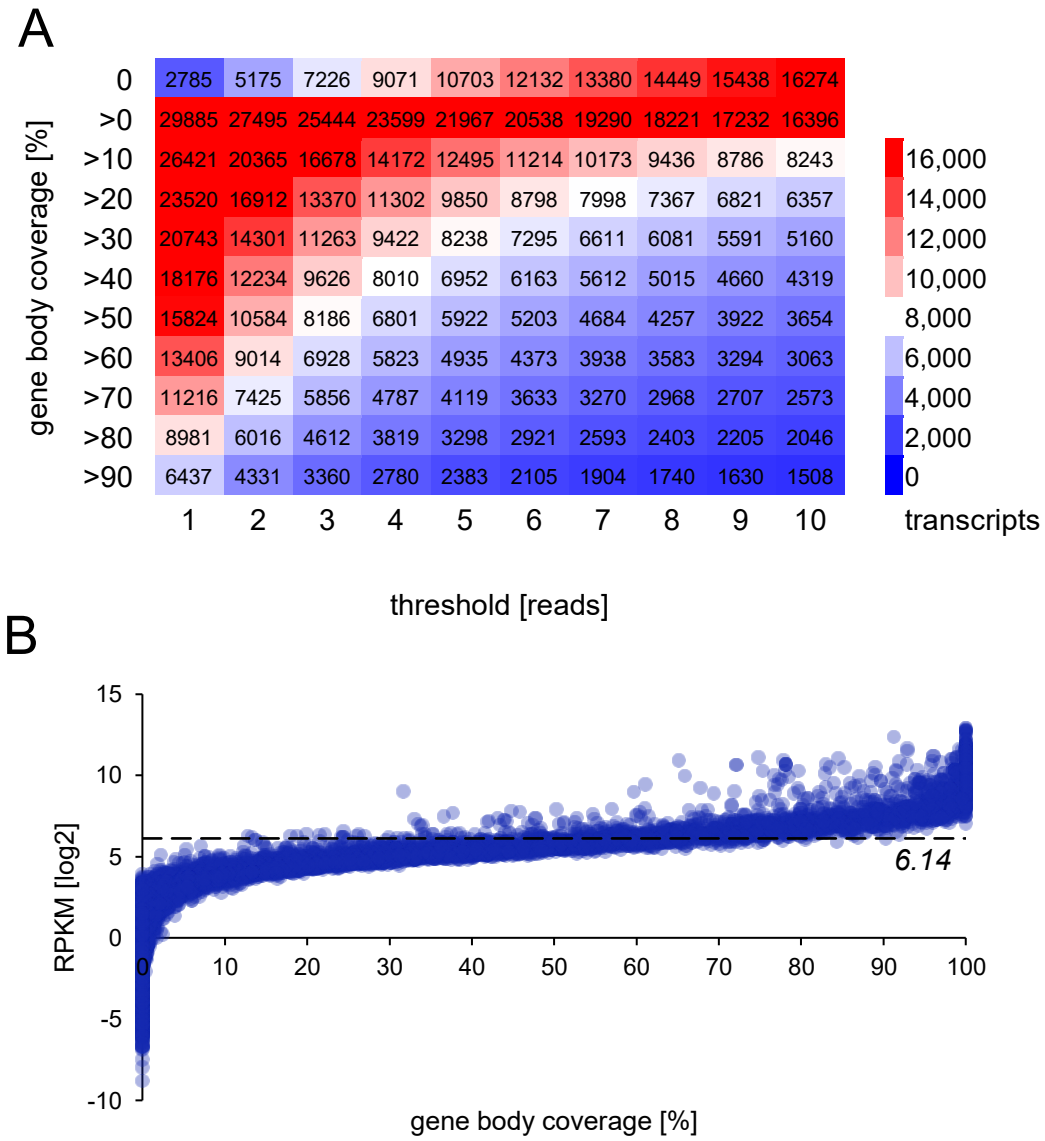


Figure 3-4: Assessment of data set depth. (A) Cumulative heat map visualizing the number of transcripts covered to a certain minimal extent (rows) by a certain minimal number of reads (columns) in sample iii (modified from Gotsmann et al., 2024). (B) Scatter plot representing the log2-RPKM values of transcripts versus their extent of gene body coverage at cut-off 10. Dashed line marks 6.14, the minimum RPKM value detected in the group of transcripts with >90% gene body coverage.

identified. However, the lower border of the dynamic range of RPKM values in the >90% category was only 6.14, as marked by the dashed black line. This line clearly visualizes that a wide range of values within this category is present throughout the whole data set, even among the transcripts that have a gene body coverage lower than 20% at this cut off. This demonstrates how categorization of the transcripts according to their gene body coverage provides a much better estimate of “completeness” of a data set than pure expression values or the number of detected transcripts alone. With this approach, the depth of a data set can be estimated in multiple dimensions according to the specific purpose of the data and interesting transcripts can be identified easily by combining their gene body coverage and expression values.

Comparing the three data sets in respect to their depth, sample iii again showed the best metrics (see Figure 3-4 A and Supplementary Figure 1) with having the most transcripts covered with >50% gene body coverage at cut-off 1 (good degree of “completeness”), having the most transcripts detected at cut-off 10 (good basal coverage) and most transcripts with >90% gene body coverage at cut-off 10 (most information content about translational dynamics).

Overall, the quality of data sets arisen from three different Ribo-Seq protocols was assessed both on technical level in terms of read losses during the read processing pipeline as well as on the biological level by RPF length- and biotype distributions and frame preference. Based on these quality parameters, protocol iii was found to produce data of the best quality, especially regarding chloroplast frame preference. However, it was found that varying frame preference across data sets might not necessarily have a negative impact on their comparability on the level of expression values, as RPKM values correlate very well between all three samples. The three protocols compared differed only in the conditions of the nuclease digest but were identical regarding the sampling procedure. The high correlation of the data sets on the level of expression values alone indicates that the impact of different digest conditions might be limited to the frame preference due to the production of different RPF length distributions that influences the accuracy of 5’-P-site- and 3’-A-site-offset calculation. Although a high in-frame preference is generally desirable, this suggests that even lower quality datasets might be sufficient for bare quantification of the translatome. Moreover, an analysis was provided for quality assessment of a data set that takes the complexity of Ribo-Seq data into account and helps to identify not only highly translated transcripts, but also the transcripts that provide the most information about their translational dynamics. However, depending on the environmental conditions the sampling culture was exposed to, not every transcript should be expected to be translationally active. Therefore, especially the detected number of transcripts might be highly variable across different environmental conditions or cell cycle stages.

3.1.3 The mixotrophic translatome of *C. reinhardtii* is dominated by anabolism-related transcripts

To analyze the *C. reinhardtii* translatome exemplarily, the 150 translationally most active transcripts in respect to their RPKM expression values were identified by averaging values of samples i to iii (Figure 3-5). The average RPKM values of these transcripts range from approximately 1,300 to 8,000 with only the top 16 transcripts having values higher than 4,000. Due to these extremely high values, these 150 transcripts only account for roughly 30% of the three data sets' total RPFs, respectively (i: 29.3%, ii: 30.5%, iii: 29.7%). Calculating the cumulated share of RPFs for every dataset along the list of transcripts in order of descending RPKM values shows that indeed only a fraction of the annotated transcripts (here 17,694 transcripts annotated as the main transcript of their encoding gene were considered) account

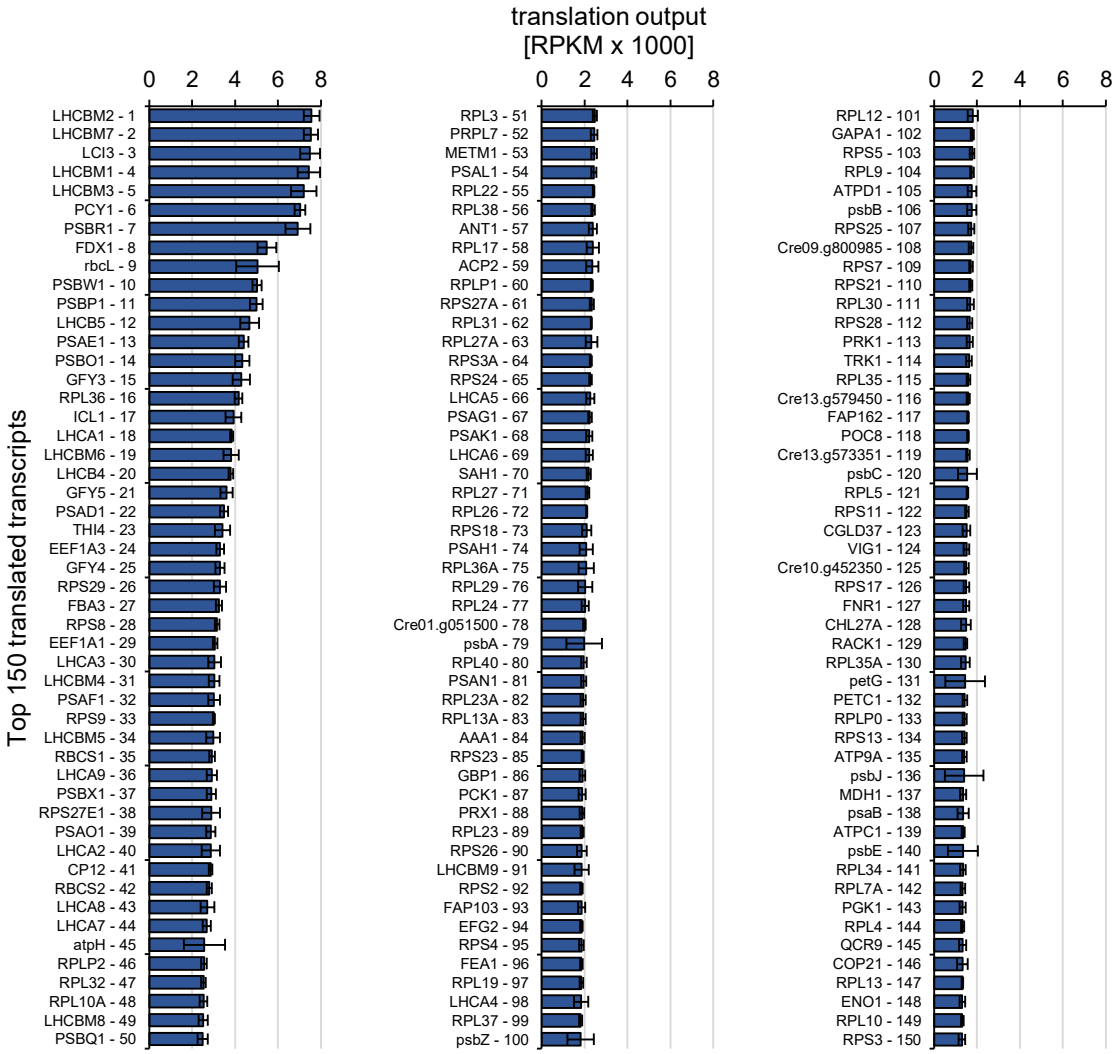


Figure 3-5: Top 150 translated transcripts. Bar chart representing translation output of the genome-wide 150 most translated transcripts as RPKM. Data represents averaged values of samples i – iii. Error bars represent standard deviations. Only ORFs of transcripts annotated as main transcript of their encoding gene were considered.

for the majority of RPFs (Figure 3-6 A) (lines do not reach 100% because of RPFs mapping to untranslated regions, intergenic regions and coding regions only included in alternative transcript variants). Calculating the translation efficiency (Ribo-Seq RPKM / RNA-Seq RPKM) of the top 150 transcripts shows that their high translation output does not necessarily coincide with a high transcript level (Figure 3-7), indicating that at least some of them are likely to be target to translational regulation. While most transcripts have values ranging between 1 and 2 in log space, especially the chloroplast encoded transcripts' (identifiable by lowercase lettering of their gene symbols) translation efficiency is usually much lower than the rest, pointing to excess transcript abundance, suggesting that translation is the bottleneck for biosynthesis of the encoded proteins. On the other hand, some transcripts like RPS29, RPL29 or LHCBM9 stand out due to very high translation efficiencies (5.5, 6.1, 6.8, in log space respectively), exemplifying a strong propensity for translation of these transcripts. Functional classification of the top 150 transcripts clearly demonstrates the dominance of anabolism-related transcripts among the translatome with a total of 62 transcripts being involved in cytosolic translation and 58 in photosynthesis. Interestingly, among the transcripts encoding metabolic enzymes, many reflect the mixotrophic lifestyle of the cultures used for the experiment. Since the cultures used for the experiment were grown under continuous light in acetate rich media, they metabolize acetate via the glyoxylate cycle to produce C4 acids and $NADPH+H^+$ further used in gluconeogenesis. Remarkably, transcripts encoding key enzymes of this metabolic pathway are found among the top-translated candidates (ICL1 – isocitrate lyase, PCK1 – phosphoenolpyruvate carboxykinase, MDH1 – malate dehydrogenase) (Plancke et al., 2014; Schnarrenberger & Martin, 2002). Moreover, three transcripts encoding the acetate transporters GFY3-5 were found as well as the acyl-carrier protein ACP2, involved in fatty acid synthesis (Durante et al., 2019). In contrast to the many transcripts encoding anabolism-related proteins, only two candidates were found in the list which are usually considered to be involved in catabolism: GAPA1 (glyceraldehyde-3-phosphate dehydrogenase) and ENO1 (phosphoenolpyruvate enolase). Although these enzymes are usually linked to glycolysis, these specific cases may represent exceptions. GAPA1 was found to be chloroplast-localized and to be involved in the Clavin-Benson cycle, supporting CO_2-fixation (Graciet et al., 2003). For ENO1, the cellular localization is currently unclear. However, a higher plant homolog of the enzyme was also found to be chloroplast-localized and hypothesized to provide pyruvate for anabolic processes like fatty acid synthesis (Bonaventure & Ohlrogge, 2002; Prabhakar et al., 2009), which according to the strong expression of ACP2 appears to be highly active in *C. reinhardtii* under mixotrophic conditions. Despite the overwhelming number of transcripts encoding 80S-ribosomal proteins, only one single chloroplast ribosomal protein, PRPL7 (new nomenclature: bL12c) is found in this set. In *E. coli*, the bacterial homolog bL12 is the binding site for translation factors IF2, EF-Tu, EF-G and RF3 (Hofmann et al., 2025), making this

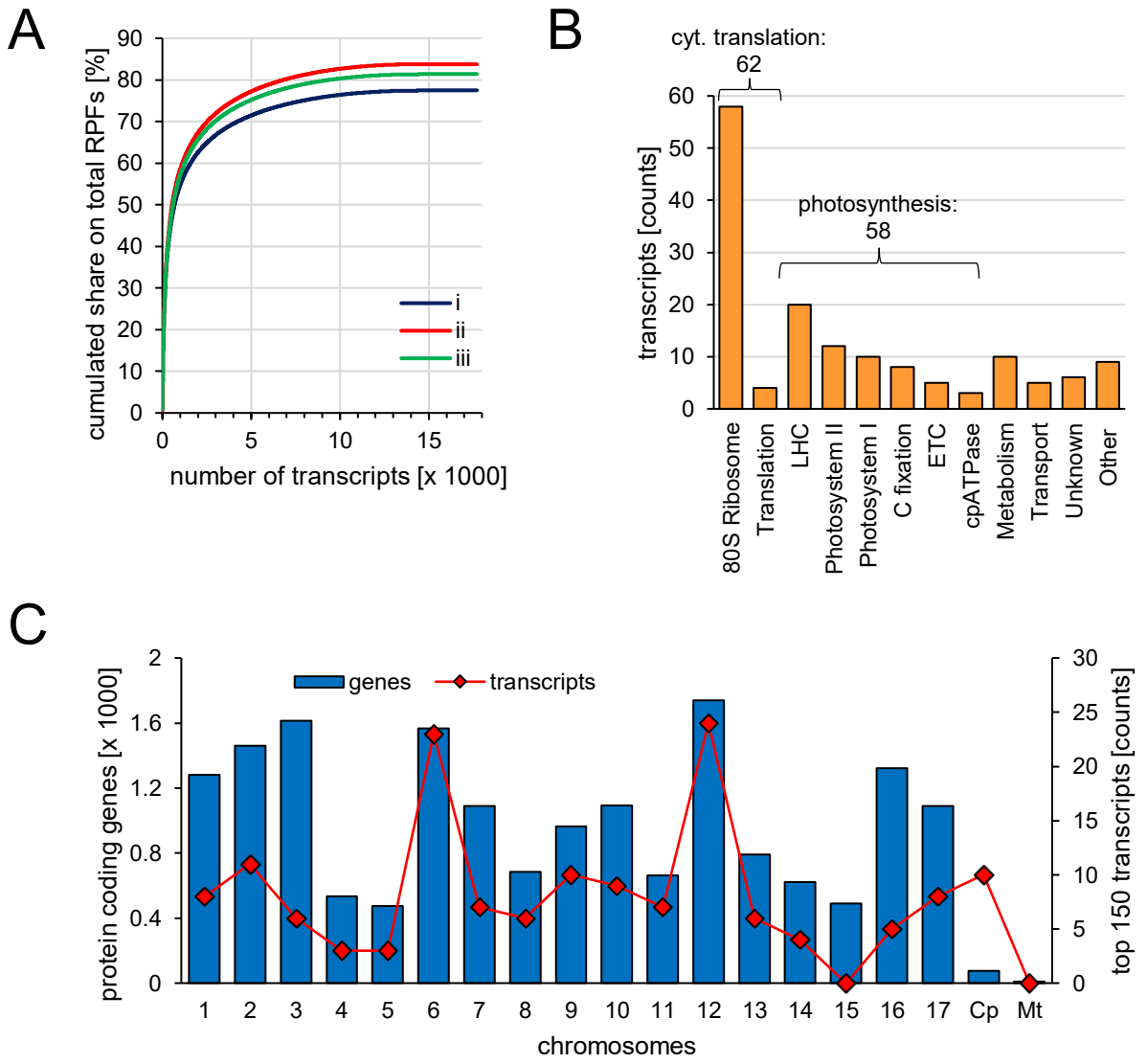


Figure 3-6: Only few transcripts account for large quantities of RPFs, many of them involved in anabolic processes (A) Line graph representing the cumulated share of RPFs of transcripts ordered by their translation output versus the respective number of summed up transcripts for every sample individually. Only ORFs of transcripts annotated as main transcript of their encoding gene were considered. (B) Functional classification of the 150 top-translated transcripts. Individual numbers: 58 80S Ribosome, 4 Translation, 20 LHC (light harvesting complex proteins), 12 Photosystem II, 10 Photosystem I, 8 C (carbon) fixation, 5 ETC (chloroplast electron transport chain), 3 chloroplast ATPase, 10 Metabolism (metabolic enzymes), 5 Transport, 6 Unknown, 9 Other. (C) Distribution of top 150 transcripts across the genome. Bars represent the total numbers of protein coding genes encoded on every single chromosome, red diamonds and lines represent the number of top 150-transcripts found on every chromosome. Only full chromosomes were considered, no scaffolds.

protein a potential candidate to regulate chloroplast translation. Given the prosperous growth conditions of the cultures used in this experiment and the mid-log phase sampling point, the strongly anabolism-centered translatome appears logical, since the cells live under virtually unlimited energy supply which strongly promotes growth and cell division, necessitating constant biosynthesis of ribosomal and photosynthetic constituents. If bL12c is a limiting factor for chloroplast translation, it would also make sense to translate it at a higher rate than other chloroplast ribosomal proteins under these conditions to eliminate a possible bottleneck in chloroplast biogenesis. Despite their high translation output, six transcripts among the top 150 encode still uncharacterized proteins (LCI3, Cre01.g051500, Cre13.g579450, POC8,

Cre10.g452350 and COP21), demonstrating the use of Ribo-Seq to find interesting new targets to study. A comparison of the chromosomal origin of the top 150 transcripts with the number of genes encoded on every chromosome (Figure 3-6 C) showed that the distribution of top-translated transcripts is comparable to the general distribution of protein coding genes across chromosomes with few exceptions: first, not a single top-translated transcript is encoded on chromosome 15, although chromosomes 4 and 5 each encoding three of the top transcripts harbor similar numbers of genes as 15. Second, although chromosomes 6 and 12 both encode a similar number of genes like chromosome 3, they encode roughly four times as many top transcripts as chromosome 3. And last, although the chloroplast genome (Cp) encodes only a fraction of the genes any nuclear chromosome encodes, it encodes a much higher portion of top transcripts, suggesting that translation output values in the chloroplast are generally higher than in the cytosol. Chromosome 15 was largely rearranged in the newest genome release due to a high content of repeats and has a considerably lower gene density than the other chromosomes (Craig et al., 2023). The reasons for this are still unclear, but it can be speculated that genes requiring a high translation output may have been evolutionarily transferred to other chromosomes. Similarly, it might be possible that translationally more active genes were sequestered to specific chromosomes. Although purely speculative, such a sequestration may potentially provide a more efficient regulation of genes that require stoichiometric expression by adding common control elements.

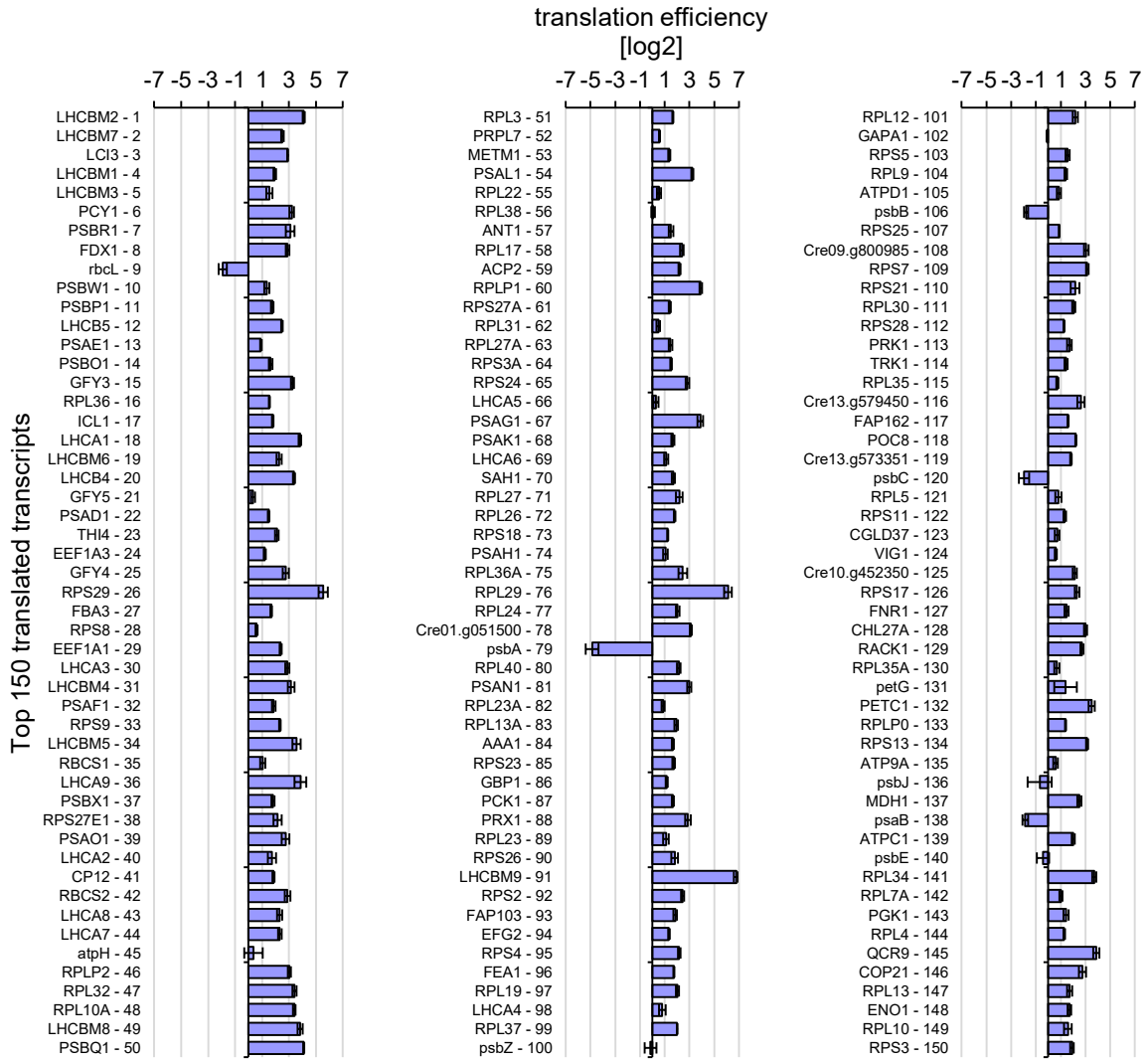


Figure 3-7: Translation efficiency of top 150 translated transcripts. Values represent transcript-wise averages of log2-transformed ratios of Ribo-Seq RPKM / RNA-Seq RPKM. Only transcripts annotated as main transcript of their encoding gene were considered.

3.1.4 Translatome-wide analysis of gene expression confirms pronounced translational control in the chloroplast and mitochondria

To get a translatome-wide overview of gene expression, all transcripts were classified by the localization of their encoded protein and their average Ribo-Seq derived RPKM values were compared to their corresponding RNA-Seq derived values (Figure 3-8, non-zero values only). Nuclear encoded transcripts range within the same scope on both axes, independently from the localization of their encoded protein product. The only notable difference between the distinct localizations may be slightly more transcripts encoding cytosolic or chloroplast targeted products among the extreme values at the upper tip of the scatter cloud than transcripts encoding mitochondrial- or ER-targeted proteins. The majority of organellar encoded transcripts in contrast display a high translation output, ranging mostly in the top half of the dynamic range. Transcripts encoded on the chloroplast or the mitochondrial genome both show a pronounced trend towards high transcript abundance with many of them representing

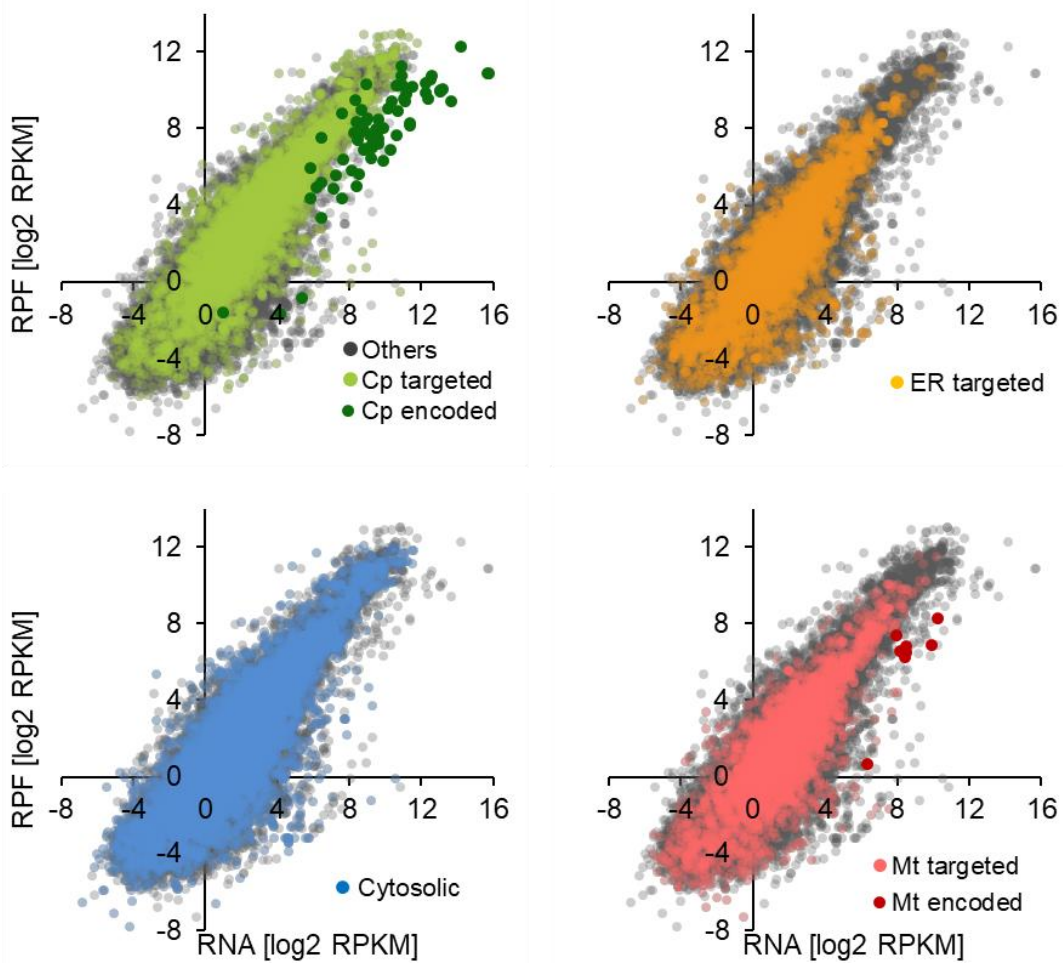


Figure 3-8: Genome-wide expression of *C. reinhardtii* transcripts on the translational level (RPF) versus transcript level (RNA) split by genetic origin and annotated localization of encoded proteins. Cp: chloroplast, Mt: mitochondrial, ER: endoplasmatic reticulum. Modified from Gotsmann et al., 2024.

the extremes on the RNA-axis, as compared to their translation output. Chloroplast transcripts have been reported to be exceptionally long-living (Klaff & Gruissem, 1991; Mullet & Klein, 1987), which might lead to these extreme accumulations of chloroplast transcripts. The comparingly low corresponding values on the translatome level indicate that translation is limiting chloroplast protein synthesis, confirming the widely accepted assumption that chloroplast gene expression is mainly post-transcriptionally regulated. Interestingly, for mitochondria-encoded transcripts, the same tendency is observable. However, translation output still correlates well with abundance of organellar encoded transcripts with Pearson's correlation coefficient of 0.83 for chloroplast transcripts and 0.81 for mitochondrial transcripts (nuclear encoded: 0.86).

Analyzing organellar gene expression in particular (Figure 3-9), it becomes evident that certain functionally related proteins are expressed at similar levels in the mitochondria (Figure 3-9 B). The mitochondrial genome of *C. reinhardtii* encodes only eight open reading frames of which five encode proteins of the mitochondrial Complex I (NADH dehydrogenase). These five transcripts form a clear cluster on the scatter plot with very narrow range in both dimensions, reflecting the stoichiometric production of the respective proteins to maintain energy homeostasis. Similar clusters also appear on the chloroplast plot (Figure 3-9 A), most prominently for the subunits of the RNA polymerase (RNAP) and ribosomal proteins and less clear for photosystem proteins (PS). Analyzing the photosystem subunits in detail (Figure 3-10 A) reveals that the four transcripts *pafI*, *pafII* (formerly known as *ycf3* and *4*), *psbN* and *psbI* towards the lower end of the RPF axis are no structural subunits of the photosystems but

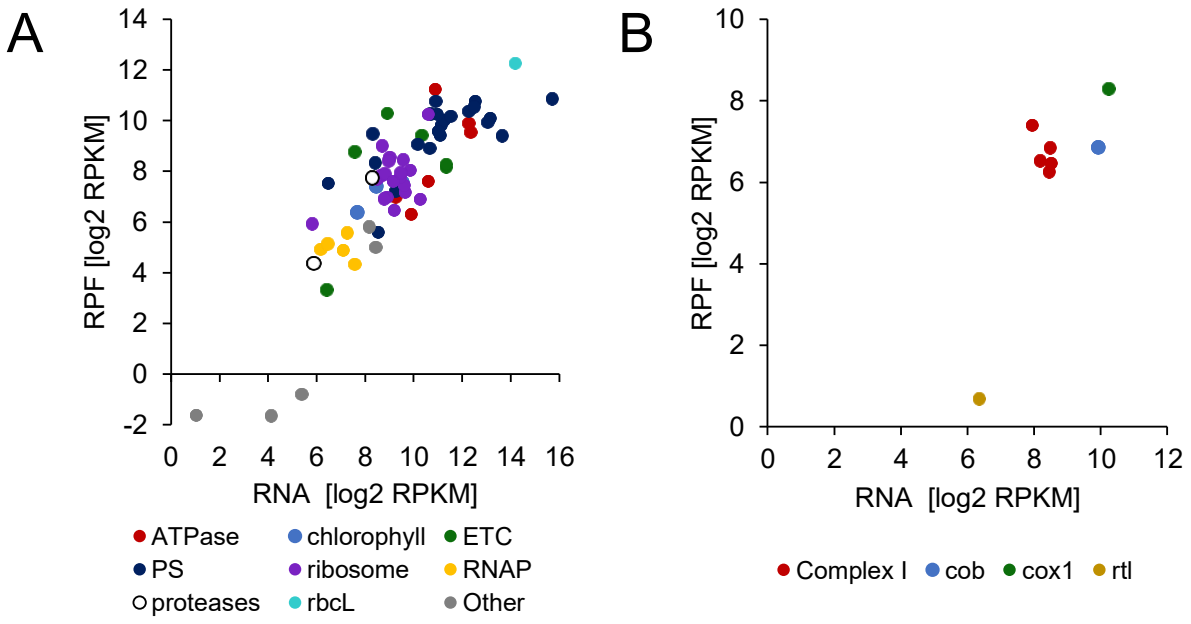


Figure 3-9: Expression of organellar encoded transcripts on the translational level (RPF) versus transcript leve (RNA). (A) Chloroplast encoded transcripts. Categorized by their funcitonal or protein complex affiliation. ETC: electron transport chain, RNAP: RNA polymerase, PS: photosystems I and II, (B) Mitochondria encoded transcripts.

most likely factors supporting the photosystem assembly (Boudreau et al., 1997; Künstner et al., 1995; Plöchinger et al., 2016). *Ycf12* instead, located at the outer realms of the cluster, was shown to represent a subunit of the photosystem II core complex, albeit not an essential one (Inoue-Kashino et al., 2011; Kashino et al., 2007). Although the exact role of these proteins is not completely clear yet, it is known that they rather play a role in fine tuning photosystem activity and therefore might be differently regulated than the essential core components of both photosystems. The remaining PS I and PS II subunits are clustering together on the translational level very well but vary considerably regarding their transcript abundance. *psbA* does not cluster together with the other PS subunits due to its enormous transcript abundance, which is the highest across the whole transcriptome. The encoded protein, D1, is an essential, but meta-stable core subunit of PS II, which in response to high light stress is quickly degraded (Aro et al., 1993). The enormous amount of *psbA* transcript might represent an adaptation of the chloroplast to quickly compensate for D1 loss in case of rapidly occurring high light exposure. This could be realized by flooding the protein synthesis machinery with *psbA*

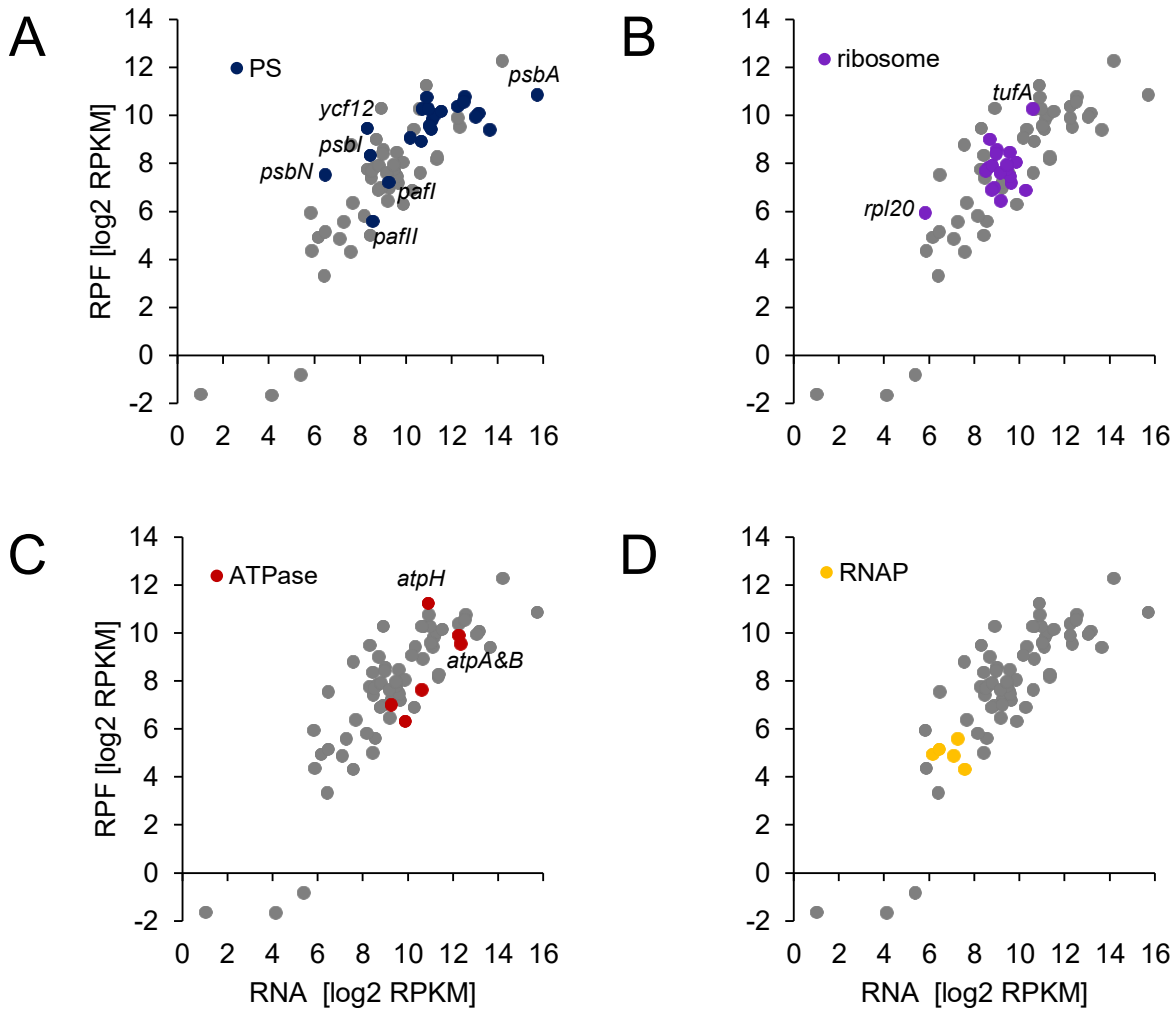


Figure 3-10: Stoichiometric expression patterns of chloroplast encoded transcripts on translational level (RPF) versus transcript level (RNA) categorized by their functional or protein complex affiliation. (A) Photosystem I and II subunits, (B) ribosomal proteins, (C) ATP synthase subunits and (D) RNA polymerase subunits. PS: photosystem subunits, RNAP: RNA polymerase subunits, ATPase: ATP synthase subunits.

transcript of a “dormant” pool. Of note, the main axis of variation for the photosystem subunits in general is their transcript abundance. Excluding *pafI*, *pafII, psbN, psbI* and *ycf12* in this context, the PS subunits vary over a range of 5.56 in log2 space (resembling over 47-fold difference in absolute numbers) along the transcript axis and even when *psbA* is also excluded, this variation still has a range of up to 3.48 (absolute over 11-fold), while the largest variation on the translation axis is only 1.94 (absolute 3.85-fold). This again suggests pronounced translational control regarding chloroplast gene expression to maintain stoichiometric synthesis of protein complex subunits despite drastically different transcript levels. However, the large spread of PS subunits regarding their transcript abundance might also point to yet uncovered mechanisms of stress acclimation, since the variation in transcript levels for other protein complex subunits seems less pronounced. Comparing PS subunits with components of the gene expression machinery reveals a striking difference between these groups (Figure 3-10 B and D). Especially the subunits of the RNA polymerase cluster exceptionally well together on both axes, revealing very limited variation. Regarding ribosomal proteins, *rpl20* and *tufA* deviate from the others. While *tufA* encodes the translation elongation factor EF-Tu and thus should not be expected to be produced in stoichiometric amounts with other ribosomal proteins, *rpl20* represents a structural component of the ribosome. However, in bacteria *rpl20* was found to play an essential role in 50S subunit assembly and is regulated via an auto-suppression mechanism by binding its own transcript (Haentjens-Sitri et al., 2008; Raibaud et al., 2003). This mechanism prevents bound *rpl20* transcript from being translated which might explain its deviation from the remaining cluster, if this autoregulation is conserved in *C. reinhardtii*. Of note, *rpl20* yields only few RPFs with a unique length distribution that all map to three specific regions within the transcript (Supplementary Figure 2). These RPFs might represent ribosomes stalled within the process of translation, due to such an auto-regulation mechanism of rpl20. Therefore, it is likely that the translation output calculated for *rpl20* is corrupted and should not be trusted. Excluding *rpl20* and *tufA* from the ribosomal cluster, the remaining transcripts hardly vary on both axes, similar to the RNAP subunits. Again, this might suggest that the pronounced variation of PS subunits on the transcript level serves a specific purpose. It is known that photosystems are organized in very dynamic and flexible supercomplexes that are constantly maintained and remodeled to fine tune and adapt photosynthesis to ever-changing environmental conditions (Minagawa, 2011). If this fine tuning fails, photosystems tend to produce reactive oxygen species that cause serious harm to the cell (Pospíšil, 2016). Considering the overwhelming number of photosystem complexes that have to be maintained simultaneously, it appears logical that accumulating large amounts of inactive transcript might lower the pressure on the system in case of an emergency when a specific subunit has to be produced quickly at a high rate to repair damaged photosystems. A limited capability to produce PS subunits can result in acute stress due to effects on the ratio

of functional PS II to PS I complexes and therefore represents a considerable threat to the cell. Ribosomes constituents or RNA polymerases in contrast may be much less maintenance-intensive than photosystems, thus such “transcript pools” as seen for PS subunits might not be necessary for them.

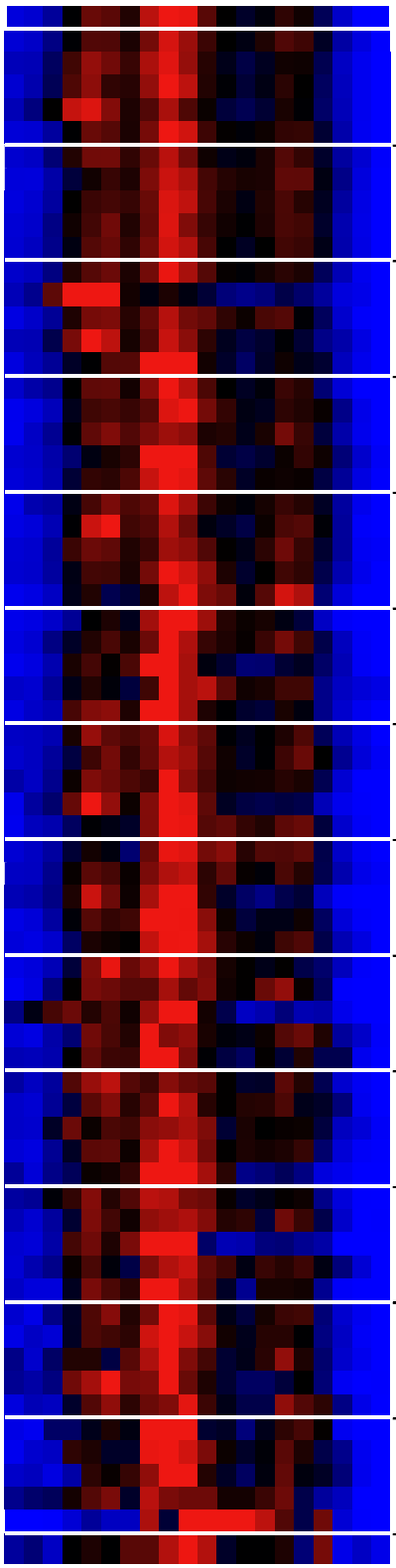
The stoichiometry of the ATPase subunits encoded on the chloroplast genome is reflected by their translation output (Figure 3-10 C). *atpH* with 14 copies per complex is the most abundant subunit and has the highest translation output (Hahn et al., 2018). Next, *atpA* and *atpB*, together forming a heterodimer that is present in three copies in the full complex both show very close expression values and are followed by the remaining *atpE, F* and *I*, which are present as monomers within the complex. This stoichiometry is also reflected very closely by the ratio of the different subunit’s translation output with an absolute ratio of 3.29 and 2.56 (theoretically: 4) between *atpH* and *atpA* and *B* as well as between *atpH* and *atpI* with 12.37 (theoretically: 14). However, the absolute ratios between *atpH* and *atpE* and *F* are much higher with 19 and 31, respectively, possibly indicating an extra layer of regulation.

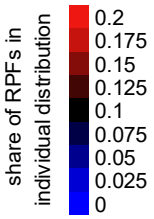
3.1.5 Chloroplast RPF length diversity may reflect different ribosomal conformations

One intriguing difference between chloroplast and mitochondrial or cytosolic RPFs is the distribution of RPF lengths. While mitochondrial and cytosolic RPFs showed very clear, approximately bell-shaped distributions, the chloroplast RPF distribution showed a tri-modal distribution with peaks at 24, 28 and 34 nt that has a strong adverse effect on P-site calculation for the chloroplast. To analyze the different RPF length species, the relative RPF length distribution was calculated for every chloroplast-encoded transcript separately that yielded more than 100 RPFs and was plotted as a heat map in Table 3-1. In general, almost all transcripts' distributions have a clear maximum in the range of 27 to 29 nt, confirming this as the chloroplast's main RPF length range. Hereafter, this RPF species is referred to as "main" species, while the other two will be referred to as "short" and "long" species. To test if the other RPF length species originate mainly from specific transcripts, an average RPF length distribution was calculated from all separately normalized length distributions, eliminating all bias from the abundance of different transcript's RPFs (Table 3-1, first row). Since the abundances of RPFs differ over a wide range among chloroplast transcripts, this equalized length distribution should differ considerably from the observed normalized distribution if a certain RPF length mainly originates from a specific set of transcripts. However, comparing these distributions, it becomes clear that the differences are minimal (Figure 3-11 A) and that all RPF length species seem to be approximately evenly distributed among all transcripts. Nevertheless, although the different length species seem to be ubiquitous among chloroplast transcripts, some transcript's distributions are clearly different from the average distribution. To quantify this difference, the root mean square error (RMSE) between each distribution and the average distribution was calculated and each transcript was ranked by its RMSE from 1 (highest RMSE) to 66 (lowest RMSE) to identify the most average and the most deviant candidates. Overall, the RMSE varies over 10-fold from the lowest (*atpH*, 0.0431) to the highest rank (*psbB*, 0.0038). Comparing the distributions of the three highest ranked candidates (*psbB*, *rps2*, *ycf1*, having the lowest RMSEs) with the average (Figure 3-11 B) demonstrates that the average distribution indeed is almost identical to these three individual distributions. Figure 3-11 C in contrast depicts the most deviant transcripts (*atpH*, *rpl20*, *petL*). Interestingly, the distributions do not only differ from the average but also from each other with *atpH* having a much lower fraction of the main and long RPF species in favor of the short species, while the distribution of *rpl20* is rather shifted towards a range between main and long (30 nt). *petL* instead shows a considerable loss of long RPFs in favor of the main species while its share of the short RPFs remains similar to the average. Of note, although the three distributions are

very different from each other, at least the distributions of *atpH* and *petL* still maintain the trimodal pattern seen in the average, even though at different peak amplitudes.

Table 3-1: Heat map depicting the distribution of RPF length species among chloroplast-encoded transcripts individually. Colors of the tiles encode the relative share of RPFs for each transcript in every length category. The table is sorted by each individual transcript's contribution to the total chloroplast ribosome pool. First line represents the average RPF length distribution. The cumulated share of Cp-RPFs represents the increasing share from summing up the individual transcript's shares along the table's index. RMSE and RMSE rank represent the root mean square error of each transcript's distribution comparing it to the average distribution and its rank among all chloroplast transcripts with 1 having the highest RMSE of all transcripts and 66 having the lowest. Only coding transcripts that yielded more than 100 RPFs were considered (*orf202*, *orf528*, *orf854* and *Icre-I* were excluded). Modified from Gotsmann et al., 2024.





		share of Cp-RPFs [%]	cumulated share of Cp-RPFs [%]	RMSE	RMSE rank
	average				
1	rbcL	19.32	19.32	0.0058	59
2	psaB	8.06	27.38	0.0073	52
3	psaA	7.30	34.67	0.0065	56
4	psbA	6.92	41.60	0.0179	12
5	psbB	6.63	48.23	0.0038	66
6	psbC	6.26	54.49	0.0090	44
7	tufA	4.24	58.73	0.0093	42
8	atpB	4.01	62.74	0.0056	60
9	atpA	3.71	66.44	0.0069	54
10	psbD	3.17	69.61	0.0052	63
11	rps2	2.27	71.88	0.0041	65
12	atpH	2.01	73.89	0.0431	1
13	rps3	1.54	75.43	0.0094	40
14	petD	1.28	76.70	0.0152	19
15	psbE	1.22	77.92	0.0132	26
16	ycf1	1.05	78.97	0.0050	64
17	psbZ	1.05	80.02	0.0084	45
18	clpP	0.99	81.01	0.0094	41
19	rpl2	0.92	81.93	0.0135	23
20	rps4	0.88	82.81	0.0065	55
21	petA	0.87	83.68	0.0064	58
22	psbH	0.86	84.54	0.0152	18
23	chlN	0.83	85.37	0.0100	37
24	rpoC2	0.82	86.19	0.0053	62
25	psbJ	0.71	86.90	0.0200	9
26	psbF	0.65	87.56	0.0245	7
27	rps8	0.65	88.21	0.0076	50
28	petG	0.64	88.85	0.0253	6
29	psbK	0.64	89.49	0.0131	27
30	petB	0.62	90.12	0.0100	38
31	rpoC1	0.58	90.70	0.0073	51
32	chlL	0.53	91.23	0.0093	43
33	ftsH	0.52	91.75	0.0077	48
34	psaC	0.52	92.26	0.0153	17
35	psbL	0.51	92.77	0.0130	28
36	rps9	0.48	93.24	0.0143	20
37	chlB	0.45	93.70	0.0077	49
38	psaJ	0.44	94.13	0.0190	11
39	atpI	0.41	94.55	0.0080	47
40	rps18	0.40	94.94	0.0111	33
41	rps7	0.34	95.28	0.0116	32
42	rpl36	0.33	95.61	0.0128	29
43	psbT	0.31	95.92	0.0272	4
44	rpl16	0.31	96.23	0.0133	24
45	psbM	0.30	96.52	0.0159	15
46	pafI	0.27	96.80	0.0135	22
47	rpl5	0.27	97.07	0.0082	46
48	rps11	0.27	97.33	0.0117	31
49	rpoB2	0.27	97.60	0.0053	61
50	ycf12	0.22	97.82	0.0264	5
51	atpF	0.22	98.04	0.0109	34
52	rpoA	0.20	98.24	0.0096	39
53	petL	0.18	98.42	0.0342	3
54	rps19	0.17	98.59	0.0104	36
55	cemA	0.17	98.76	0.0154	16
56	rpoB1	0.16	98.92	0.0064	57
57	rps12	0.15	99.07	0.0073	53
58	rpl14	0.15	99.22	0.0118	30
59	psbI	0.13	99.35	0.0175	13
60	atpE	0.13	99.48	0.0133	25
61	psbN	0.12	99.60	0.0232	8
62	rpl23	0.12	99.71	0.0108	35
63	pafII	0.11	99.82	0.0195	10
64	rps14	0.08	99.90	0.0135	21
65	rpl20	0.06	99.96	0.0426	2
66	ccsA	0.03	100.00	0.0164	14

This indicates again that the RPF lengths are not randomly distributed among the translatome. It is also important to mention that while the distribution of *rpl20* might be disturbed due to a low coverage (rank 65 of 66 according to RPF abundance), this is very unlikely for *atpH* which is expressed at a high level (abundance rank 12). Even *psbA*, having the fourth highest RPF abundance, shows a deviant distribution (RMSE rank 11) that is strongly skewed towards the short RPF species (Figure 3-11 D). For example, such deviations are also seen for photosystem subunits *psbJ* and *psaC* with the latter one exhibiting a similar pattern like *psbA*, while *psbJ*'s distribution is rather skewed towards the long RPF species. Interestingly, the main peak of *psbJ*'s distribution (29 nt) has a small shoulder at 31 nt, which, together with the distribution of *rpl20* (Figure 3-11 C), that has a peak at 30 nt, might hint to a fourth, underrepresented RPF species that may be masked by the others. Since the different RPF

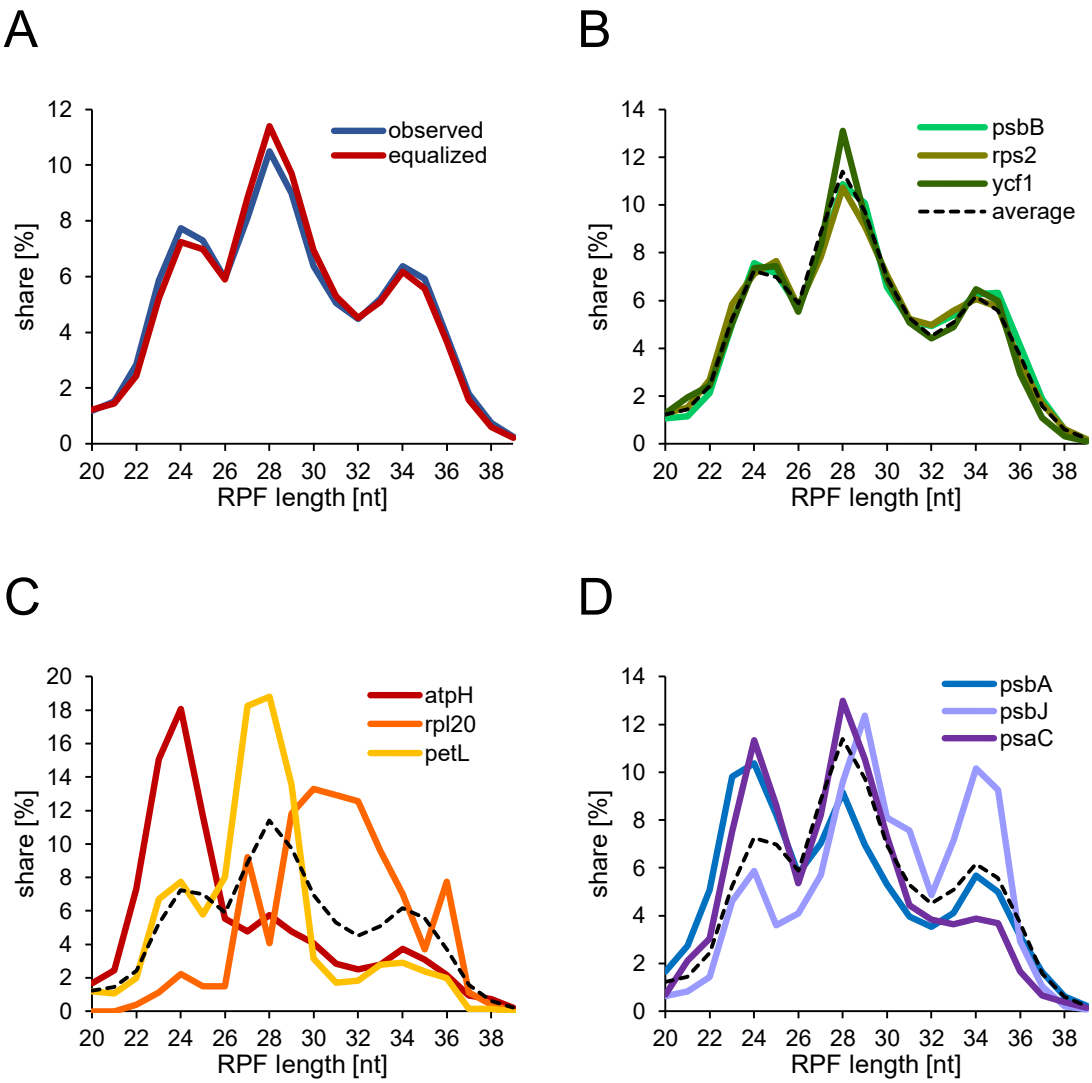


Figure 3-11: Line plots representing the relative distribution of RPF length species of chloroplast-encoded transcripts. (A) Representation of the overall RPF length distribution as percentage share of all chloroplast RPFs (observed) and average of all intra-transcript normalized distributions, simulating an equal RPF abundance for all transcripts (equalized). (B) Relative RPF length distribution of the three transcripts with lowest root mean square error. (C) Relative RPF length distribution of the three transcripts with highest root mean square error. (D) Relative RPF length distribution of the three specific transcripts *psbA*, *psbJ* and *psaC*. A, C and D modified from Gotsmann et al., 2024.

species cannot be attributed to certain transcripts, it can be assumed that they rather reflect different states of the ribosome that are connected to different structural conformations or complexes with interacting factors, thus changing the footprint of a ribosome on the transcript molecule. To explore this hypothesis, the average length of RPFs covering every nucleotide position of an ORF was calculated and compared to the so-called “ribosome profile” of that ORF (meaning the distribution of covering reads along an ORF). Interestingly, in the cases of *atpH* and *psbA*, a pronounced decline of the average RPF length can be observed at positions of their ORFs that have an exceptionally high coverage (Figure 3-12), indicating that the distribution of RPF lengths at these positions (*atpH*: codons 3 – 12, *psbA*: codons 141 – 150) change towards the shorter species. Although the phenomenon is especially striking in the cases of *psbA* and *atpH*, in other transcripts it is less pronounced and even in these two cases, a general correlation between the coverage of a position of an ORF and the average RPF length at that position cannot be confirmed (Supplementary Figure 3). Of note, the accumulations of RPFs in the profiles of *psbA* and *atpH* in relation to their remaining coverage are extreme cases that are rarely observed among other chloroplast encoded transcripts, thus the effect may be better observable for these two. Generally, high coverages at specific sites of an ORF are seen as indicators of a ribosomal slowdown or pausing that leads to a higher chance of sampling RPFs at these positions. If such sites coincide with shorter RPF length, this suggests that ribosomes may acquire a specific preferred conformation during a slowdown or pausing event, yielding a shorter footprint. High Pearson correlation coefficients for the ribosome profiles of *atpH* (i/ii: 0.91, ii/iii: 0.90, iii/i: 0.95) and *psbA* (i/ii: 0.95, ii/iii: 0.87, iii/i: 0.93) (Supplementary Figure 4 B and D) between the different samples indicate that they indeed reflect translation dynamics of the transcripts. Interestingly, even if coverage and average RPF length of a specific position within an ORF may be independent from another, the distribution of average RPF lengths along the *psbA* transcript shows surprisingly high Pearson correlation coefficients between the different samples (i/ii: 0.62, ii/iii: 0.74, iii/i: 0.60), suggesting that also the average RPF length may reflect transcript specific translational features, at least in some cases (Supplementary Figure 4 C and A for comparison). Given the complicated orchestration of the synthesis of different protein complex subunits within the chloroplast that also involves the assembly of subunits of different genetic origin, it is conceivable that translational slow down events are widespread in the chloroplast, explaining the great abundance of short RPFs. It is known that expression of the *psbA*-encoded D1 protein, a reaction center subunit of photosystem II, is stringently regulated through a CES cascade (control by epistasis of synthesis) (Minai et al., 2006; L. Zhang et al., 1999), a mechanism in *C. reinhardtii* chloroplasts that ensures the stoichiometric synthesis of subunits that belong to the same protein complex. In this cascade, synthesis of the D1 protein is regulated by the availability of its complex partner D2 (*psbD*). The *psbA* ORF shows a large accumulation of RPFs covering codons 141 to 150

which also displays the pronounced decline in average RPF length (Figure 3-12 B). Taking the length of the polypeptide exit channel into consideration which shields 30 to 40 amino acids (Chadani et al., 2021), the second transmembrane domain of D1, which also binds a chlorophyll molecule, is about to emerge from the tunnel when this region of the ORF is translated. It is thinkable that translation is paused at this region to ensure transmembrane domain incorporation into the thylakoid membrane or chlorophyll binding before the second TMD emerges from the ribosome. Otherwise, the two TMDs (and the following) could possibly interact with each other in an undesired way and cause the protein to misfold and aggregate. Similar mechanisms are also thinkable for other proteins since most chloroplast encoded proteins are complex subunits which might require pronounced translational regulation as well.

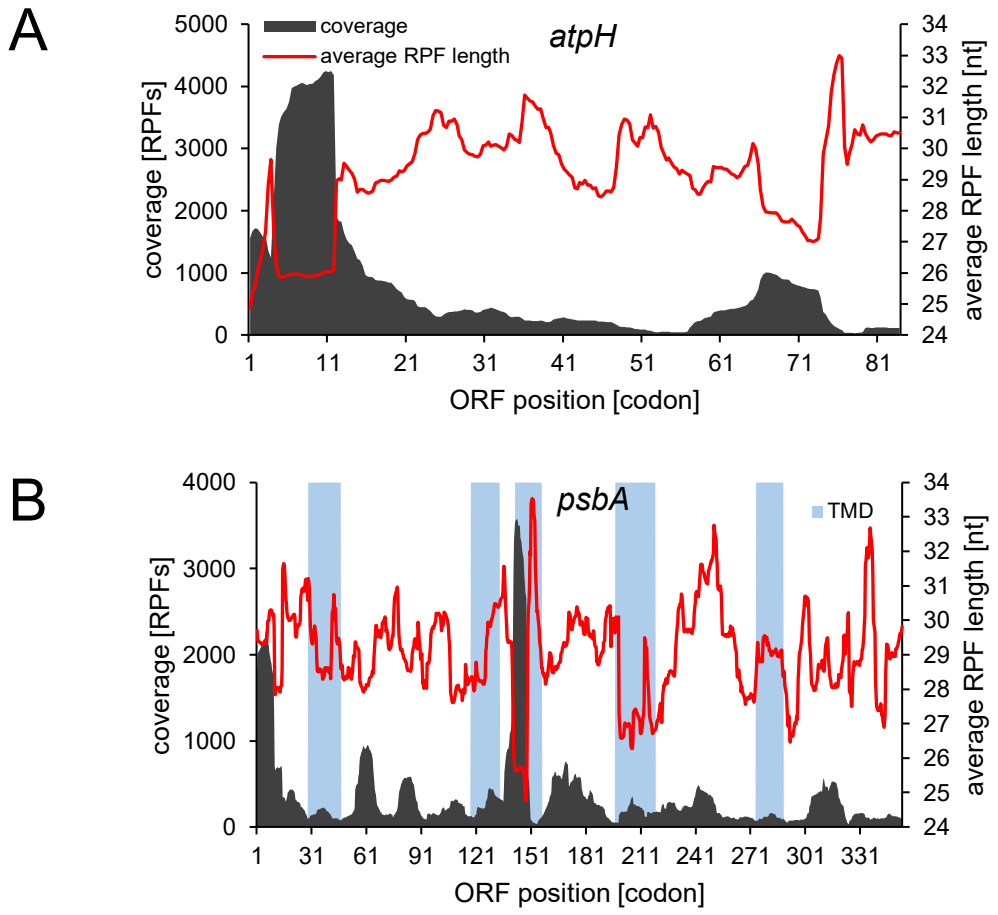


Figure 3-12: Plots representing the distribution of RPFs and average RPF length along chloroplast-encoded transcripts *atpH* (A) and *psbA* (B). Grey areas represent coverage and refer to the left y-axis, read lines represent the average readlength at a position and refer to the right y-axis. Blue-shaded areas in B mark the regions of the psbA transcript that encode transmembrane domains (TMD). Modified from Gotsmann et al., 2024.

While the accumulation of shorter RPFs seem to coincide with translational pausing or slowdown, the origin of the longer RPFs is less clear. Although the sudden decrease of average RPF length in *psbA* seems to be followed by a sudden increase in RPF length, such spikes can also be observed on other occasions without a specific accompanying event in the ribosome profile. Considering the increase in RPF length compared to the main RPF species,

it is thinkable that the larger species may be the result of the binding of ribosome associated factors extending the footprint. The reason for the shorter RPF length could be an arrest of the ribosome in a partly rotated state during translocation that provides the footprint less protection from digestion as it was observed in yeast (Lareau et al., 2014).

3.1.6 Ribosome profiles reproducibly reflect transcript-specific translational dynamics

Ribo-Seq data does not only provide expression values for transcripts, as seen before in Figure 3-12. Instead, the distribution of coverage along an ORF provides a much deeper insight into the translation of specific transcripts than an expression value alone could provide. It is known that translation speed can vary within one ORF due to various reasons like the occurrence of co-translational events, tRNA abundances or secondary structures in the transcript molecule, to name only a few (Uematsu & Qian, 2025). Since the dwell time of a ribosome at one specific site of a transcript molecule is proportional to the chance of sampling a ribosome in this position, the distribution of coverage along a transcript is a function of translation speed, ultimately mapping the translational dynamics of the transcript. Positions covered by especially high peaks within a ribosome profile are usually interpreted as sites of ribosomal slowdown or even pausing which may be of specific interest. However, in order to interpret ribosome profiles, it is important to verify their reproducibility first. Due to the nature

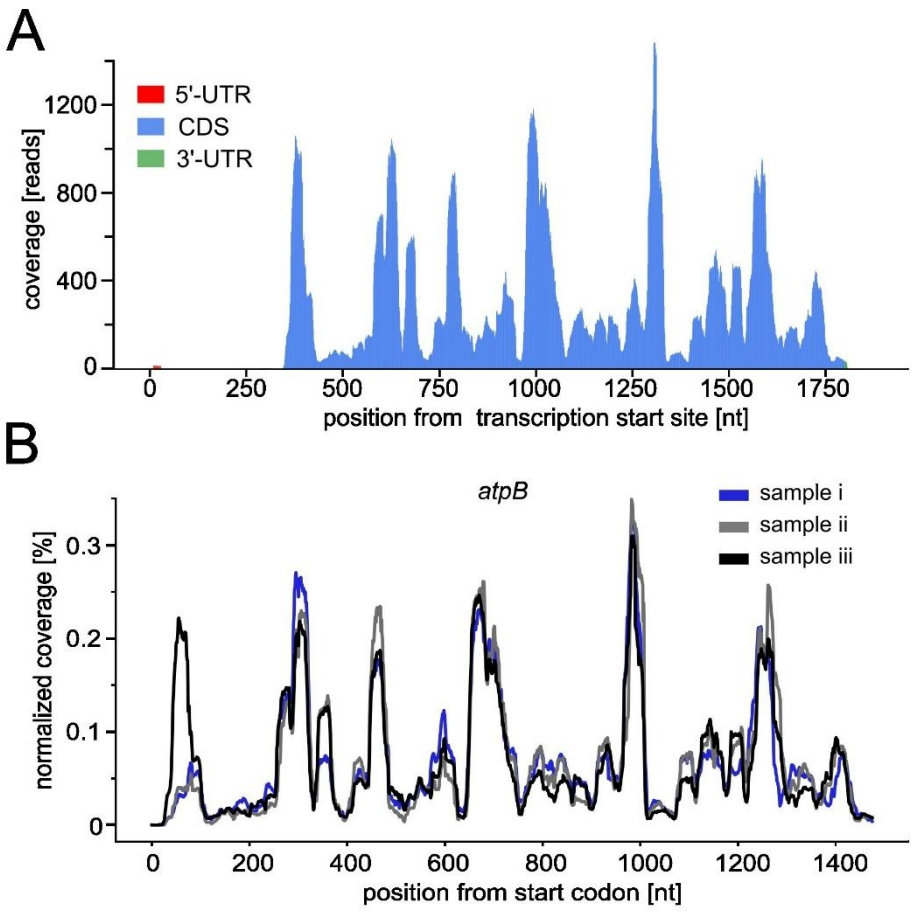


Figure 3-13: Representative examples of ribosome profiles. (A) Single ribosome profile of atpB from sample iii containing UTR regions. (B) Overlay of normalized *atpB* ribosome profiles from samples i to iii. Profiles were normalized by dividing each positions read coverage by the sum of all positions read coverage and calculating percentage values.

of Ribo-Seq, the UTRs of a transcript should be virtually free of any RPFs while the coding sequence should be highly covered by RPFs as exemplified by the ribosome profile of the chloroplast encoded ATPase subunit *atpB* in Figure 3-13 A. Although the three different digest methods yielded notable differences in tri-nucleotide periodicity, the ribosome profiles of *atpB* exert a remarkable similarity between all three samples (Figure 3-13 B). To screen genome-wide for the reproducibility of ribosome profiles, the ribosome profiles of all annotated transcripts were compared by calculating Pearson's correlation coefficient for every transcript's ribosome profile between all sample pairs. Since a similarity between two profiles could also be the consequence of sequencing bias rather than true translation dynamics, every ribosome profile was compared to its coverage profile of the samples' corresponding RNA-Seq data set which should also reflect sequencing bias but not translation dynamics. As displayed in Figure

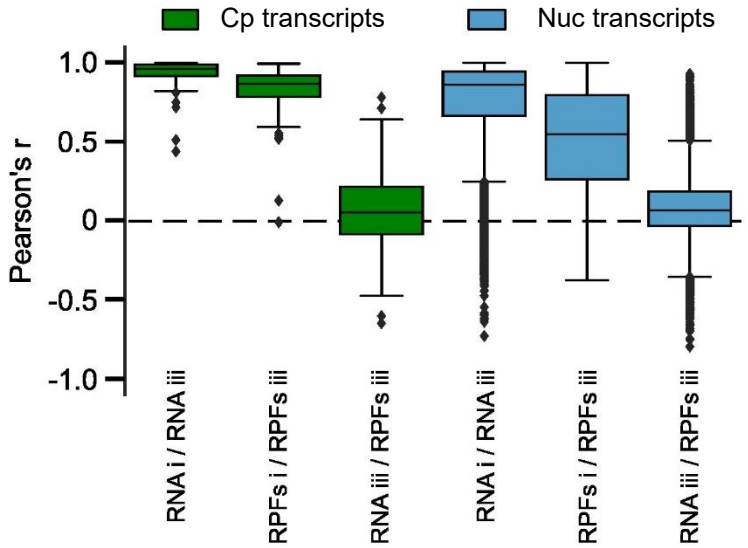


Figure 3-14: Correlation of ribosome profiles and RNA-Seq coverage profiles between samples i and iii. RNA-Seq coverage profiles and ribosome profiles were each separately compared between samples i and iii while RNA-Seq coverage profiles vs. ribosome profiles were only compared between data sets of sample iii alone. Cp: chloroplast, Nuc: nuclear.

3-14, the correlation coefficients calculated between the RNA-Seq and Ribo-Seq data sets of samples i and iii (see Supplementary Figure 5 for an overview of all sample pairs) each mainly range above 0.5 for chloroplast encoded transcripts, generally indicating a high correlation between samples for RNA-Seq coverage profiles and ribosome profiles separately. However, the correlation coefficients between RNA-Seq coverage profiles and corresponding ribosome profiles within sample iii range approximately from -0.5 to 0.5 with a median value close to zero. Although the range of correlation coefficients of the cytosolic profiles is wider than for the chloroplast profiles, the same trend can be observed for cytosolic transcripts. This indicates that RNA-Seq coverage profiles and ribosome profiles for themselves display a good reproducibility, but the underlying reasons are presumably different. Therefore, sequencing

bias alone is an insufficient explanation for the robustness of ribosome profiles between samples. Despite many transcripts' ribosome profiles showing a good reproducibility between samples, a large fraction of ribosomes profiles was less reproducible. However, comparing the

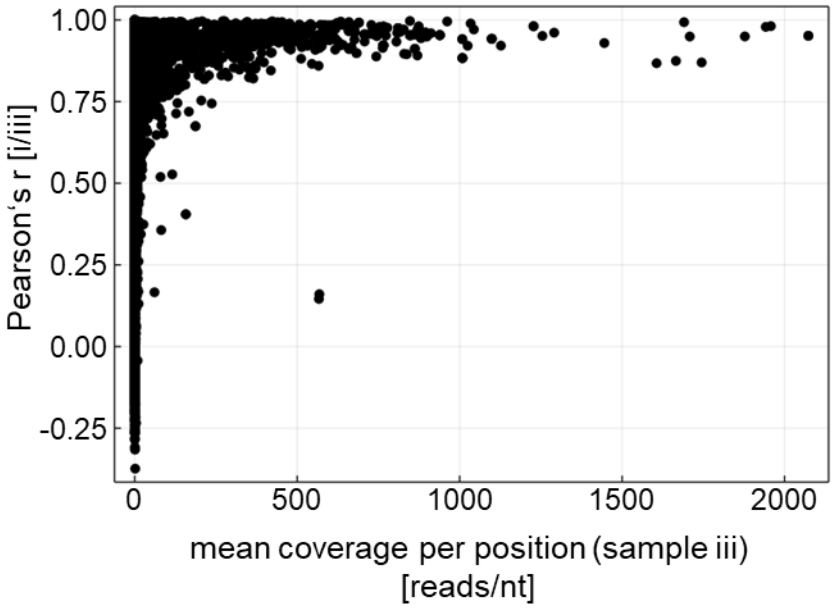


Figure 3-15: Transcript-wise Pearson's correlation coefficient of ribosome profiles between samples i and iii compared against coverage in sample iii. Average coverage per position was calculated by dividing the total number of reads mapping to a transcripts ORF in sample iii by the length of that ORF.

Pearson correlation coefficients calculated between samples i and iii with the average coverage per nucleotide of each transcript (Figure 3-15, see Supplementary Figure 6 A and B for other samples) clearly shows that low correlation coefficients exclusively appear for transcripts with average coverages close to zero. This finding indicates that low correlation coefficients of ribosome profiles between samples may be a consequence of insufficient sequencing depth and therefore limited observability instead of lacking reproducibility. Therefore, the affected transcripts do not contradict the hypothesis that ribosome profiles reflect translation dynamics.

However, if a slowdown or pausing site is found, the reason for this event may still be difficult to identify. This is because such events could by triggered either via mechanisms acting directly on the transcript molecule or indirectly on a polypeptide segment emerging from the ribosome. A direct mechanism for example could be a protein binding to the transcript and functioning as a temporary roadblock that prevents the ribosome from translocation. In such a case, locating a slowdown or pausing event to its specific trigger would be easy since this mechanism would yield a higher RPF abundance of the region directly preceding the roadblock. However, an indirect mechanism on the other side could be triggered by a nascent chain interacting factor like the signal recognition particle in the cytosol. The 30 to 40 most C-terminal amino acids of the nascent chain are buried within the polypeptide exit channel of the translating ribosome,

shielding the respective amino acids from interaction with any non-ribosomal factors (Chadani et al., 2021). Due to this reason, the true trigger of the slowdown event in such a case gets active with a delay just when the already decoded polypeptide segment emerges from the exit channel yielding an accumulation of footprints 30 to 40 codons downstream. This adds an extra layer of complexity and uncertainty to the mapping of slowdown or pausing events and the position of their respective trigger in the transcript sequence. It could even be possible that triggers act far away of their encoded position via the three-dimensional fold of the transcript molecule and may therefore be completely unmappable to the RPFs they yield.

3.1.7 Ribosome profiles help to refine genomic annotation

Due to the nature of translation, Ribo-Seq data sets mainly contain coverage within the coding region of a transcript. This, and the circumstance that RPFs usually accumulate to a larger extent over start codons can be exploited to refine the annotation of a transcript. To date, most genomic annotations are based on algorithms that predict promoters, open reading frames as well as splicing sites by reference to the genomic sequence of an organism. These predictions are mostly supported by transcriptomics and proteomics analyses to refine the genome and verify correct predictions. However, while transcriptomics approaches may be efficient in identification of transcribed genomic regions and their splicing sites, ORF verification via proteomics suffers from the indirect evidence it yields and from the technical limitations of common proteomics workflows. First, the detection of peptides encoded by a proposed ORF can be problematic if this peptide is also found within proteins encoded by other ORFs. Even if the sequence encoding a specific peptide differs between multiple ORFs, due to the degeneracy of the genetic code it is impossible to determine the origin of this peptide. Second, common proteomics workflows rely on the tryptic digest of proteins to yield peptides that can be analyzed via MS. However, trypsin reliably cleaves peptide bonds C-terminally of the basic amino acids arginine and lysin. This is problematic for proteins or domains rich in basic amino acids (for example ribosomal proteins or RNA-binding domains) that often yield very small peptides that cannot be assigned to one protein encoded by a specific ORF. In these cases, proteomics is only capable of incomplete verification of predicted ORFs, necessitating another method to complete the verification. Ribo-Seq derived reads allow a gapless coverage of ORFs if expression on the translational level and sequencing depth are sufficient. By manual inspection of the data of sample iii, multiple transcripts were found with misannotated start codons, even among highly translated transcripts whose protein products are well known to the scientific community, as exemplified by the ribosome profiles of *RPL3* (cytosolic 80S ribosomal protein L3) and *LHCA4* (light harvesting protein of photosystem I A4) in Figure 3-16 A and B. In both cases, the annotated start codon is upstream of the start of the covered region. Also, in both cases coverage starts with a clear peak covering an AUG-codon, like it is usual for a translation initiation site. Both transcripts are well translated with very high RPKM values (*RPL3*: 2264.4, *LHCA4*: 1653.4), indicating that it is unlikely that the missing coverage is attributable to a lack of sequencing depth. Moreover, in both cases the annotated alternative transcript variants are unable to explain the gap. Accepting the newly identified translation start sites shortens the LHCA4 protein by 82 amino acids that account for 9.26 kD, roughly 24% of the annotated proteins amino acids positions (346) and mass (37.94 kD). In case of RPL3 the difference is less dramatic with a loss of 23 N-terminal amino acid positions and 2.39 kD mass,

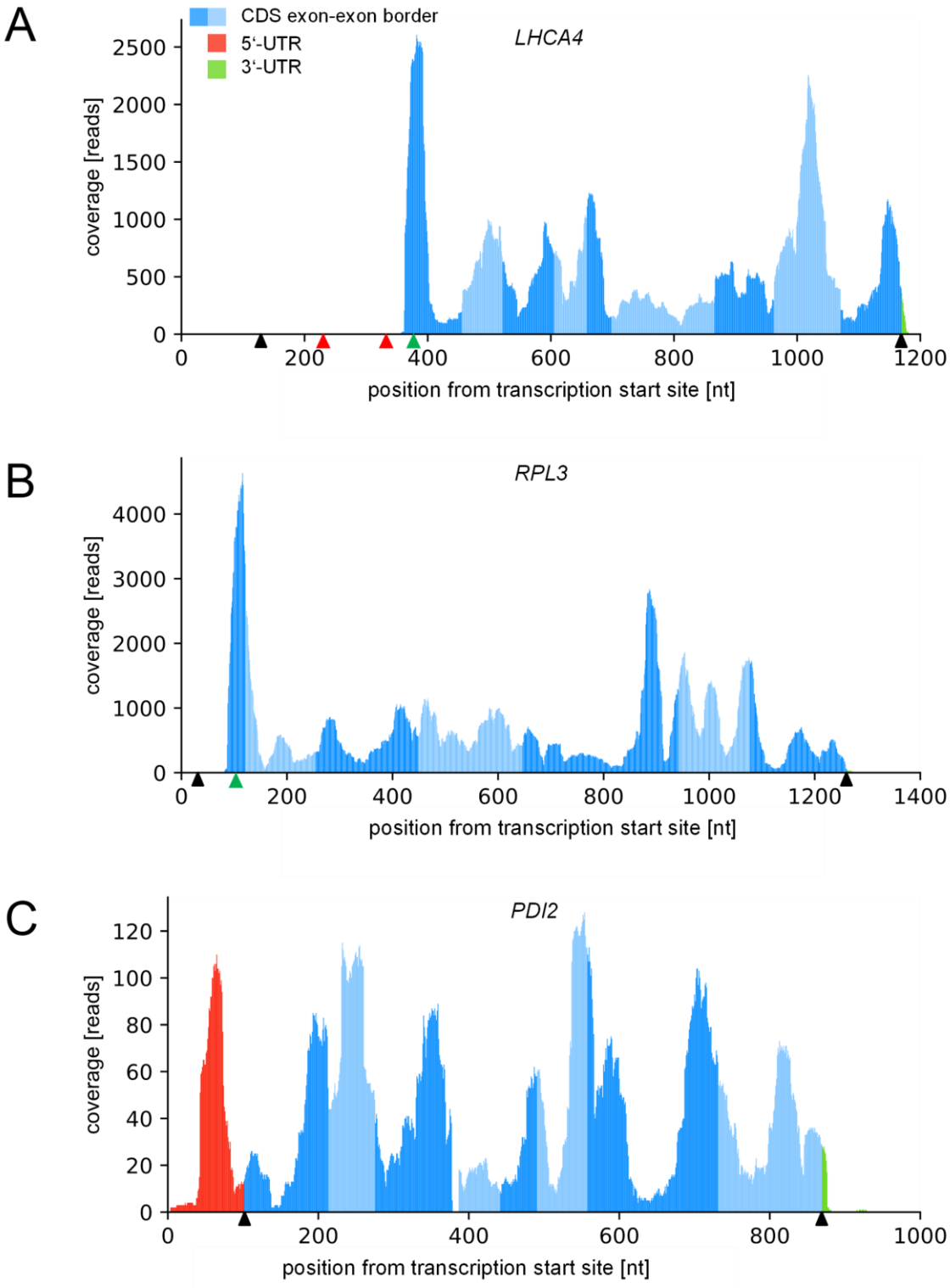


Figure 3-16: Ribosome profiles of presumably misannotated transcripts. (A) Profile of *LHCA4*. (B) Profile of *RPL3*. (C) Profile of *PDI2 (transcript 1)*. Black arrows indicate annotated start and stop codons, red arrows indicate skipped in-frame start codons and green arrows indicate start codons suggested by the ribosome profiles.

corresponding to roughly 5.2% of annotated RPL3's amino acid positions and 5.6% of its mass. A remarkable detail of *LHCA4*'s example is the fact that three in-frame AUG-codons (Figure 3-16 A, indicated by black and red arrows) must be skipped by the ribosomal pre-initiation complex during scanning the 5'-UTR to produce the observed ribosome profile. Checking all

four possible versions of the protein, the version with the start codon corrected according to the ribosome profile was the only version that is predicted to contain a chloroplast transit peptide by the Target-P 2.0 prediction program (Chloroplast score: 0.94) (Almagro Armenteros et al., 2019). The other in-frame AUG-codons yield mitochondrial or no prediction at all, supporting the hypothesis that the ribosome profile reflects the correct variant.

Another example of a possibly misannotated transcript is *PDI2*, a protein disulfate isomerase (Figure 3-16 C). The annotated start codon is poorly covered, raising doubts if it represents a true initiation site. Instead, a region within the 5'-UTR, shortly upstream of the annotated start codon shows a noticeable peak. Interestingly this peak does not cover a cognate start codon in any frame, but three so-called "near-cognate" start codons all in-frame with the annotated ORF (UUG, UUG and AUC, respectively, normally encoding leucin and isoleucin). Near-cognate start codons differ by only one base from the cognate start codon AUG and have been identified in several cases via coupled Ribo-Seq and mass spectrometry approaches to serve as translation initiation sites (Menschaert et al., 2013; Na et al., 2018). The ribosome profile of *PDI2* and the fact that these near-cognate start codons form one uninterrupted reading frame with the annotated ORF suggest that one of these alternative initiation sites may be used instead of the annotated start codon. However, identifying alternative initiation sites, cognate or not, requires very careful consideration and especially in non-cognate cases experimental verification. Therefore, the coverage data of sample iii was uploaded to the Phytozome websever (https://phytozome-next.jgi.doe.gov) and linked to the *C. reinhardtii* genome versions 5.6 and 6.1, respectively. This allows the community to browse this data in a genome browser and to curate annotated transcripts of interest manually, providing additional certainty about ORF annotations to researchers, especially when studying previously uncharacterized proteins.

3.1.8 Extreme coverage of start codons may indicate an underestimated extent of translational regulation in the cytoplasm

A common feature of eukaryotic Ribo-Seq data is the accumulation of RPFs within the close range of start codon, called “initiation peaks” hereafter. These peaks can be considered a consequence of initiation being much slower than elongation, thereby increasing the probability to sample ribosomes covering a start codon (Rodnina, 2018). An interesting detail of the data of this study is the varying height of these initiation peaks even among transcripts with similar expression strength and functional relationship, as exemplified by ribosome profiles of *PSRP7* (plastid specific ribosomal protein 7) and *PRPS5* (plastid ribosomal protein S5) in Figure 3-17. Both proteins are subunits of the chloroplast 30S ribosomal subunit and are translated similarly well according to their RPKM values. However, the initiation peak of *PSRP7* is considerably higher than that of *PRPS5*. By manual exploration of ribosome profiles, it was observed that initiation peaks usually start 12 to 13 nucleotides upstream of the start codon (as expected due to the ribosomal 5’-P-site offset, see paragraph 3.1.9) and cover the first five to seven codons of an ORF, also reflected in the metagene coverage plot in Figure 3-18 B. After codon six, metagene coverage rapidly decreases. Taking the typical length (30 nt) and 5’-P-site offset (12 nt) of cytosolic RPFs, coverage in the first six codons (corresponding to a length of 18 nt) may indeed be attributable mainly to ribosomes having the start codon bound to their P-site. Together with the rapidly decreasing coverage after codon six, this indicates that these peaks truly are artifacts of the slow initiation process. To analyze this phenomenon in greater detail, the average coverage per nucleotide of the first seven codons and of the remaining ORF were calculated for every annotated transcript and the ratio of both averages was formed (13.2 for *PSRP7*, 0.76 for *PRPS5*). Transcripts having a ratio greater than four were classified as extreme candidates. Since transcripts with a gapped coverage profile may tend to yield a disproportionately higher ratio due to a low average coverage downstream of the initiation region, only transcripts were considered, that were covered to an extent of more than 70% for this analysis to avoid such bias. Roughly 38% of considered cytosolic transcripts (3,346 of 8,757) were extreme candidates while only 11% of considered chloroplast encoded transcripts (7 of 66) and not a single mitochondrial transcript (0 of 7) showed such an extreme ratio. Also, the range of the ratios differed drastically between nuclear and chloroplast encoded transcripts (Figure 3-17 D) with 50% of nuclear transcripts having a ratio greater than 2.72 and the top 10% having a ratio greater than 11.18, while for chloroplast encoded transcripts, these values were 1.01 and 4.16, respectively. The maximum ratio that was observed was 92.4 for *GLN4*, a moderately translated nuclear encoded transcript that encodes a glutamine synthase (RPKM 39.16). The maximum values observed for chloroplast encoded transcripts all ranged between

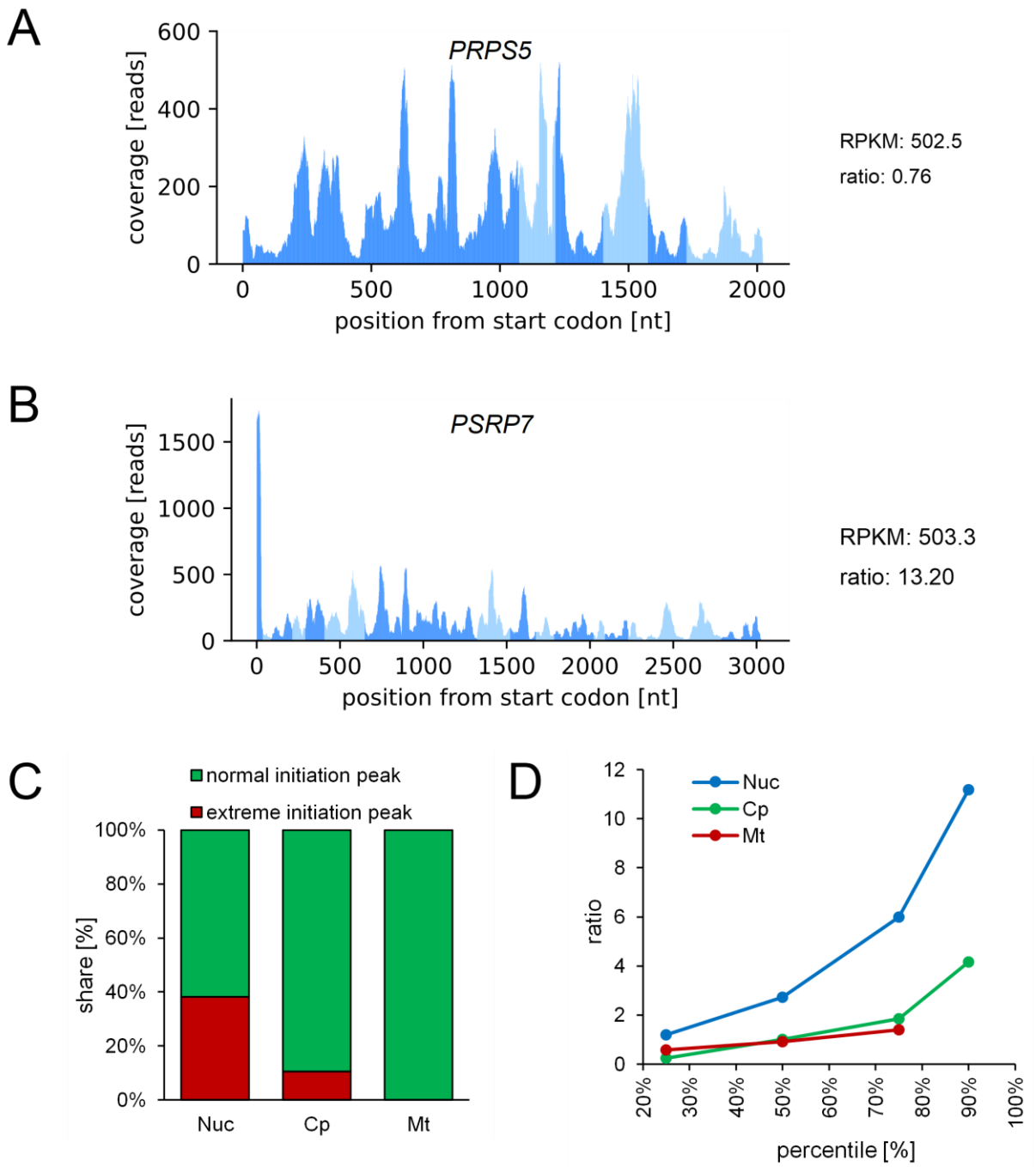


Figure 3-17: The amplitude of initiation peaks differs greatly, even between functionally related and equally well translated transcripts. (A, B) Ribosome profile of plastid ribosomal protein S5 (*PRPS5*) and plastid specific ribosomal protein S7 (*PSRP7*) from sample iii. Different exons within the ORF are marked in alternating shades of blue. (C) Column chart representing the share of transcripts having extreme initiation peaks among considered transcripts encoded in the three *C. reinhardtii* genomes. (D) Graphs representing the 25th, 50th, 75th and 90th percentile borders of the average coverage within the first seven codons over the average coverage within the remaining ORF for all three genomes. Nuc: nuclear encoded transcripts, Cp: chloroplast encoded transcripts, Mt: mitochondrial encoded transcripts. 90th percentile border of mitochondrial encoded transcripts was not calculatable due to the low number of transcripts.

6.0 and 6.75 for the photosystem II subunits *psbA*, *psbH*, *psbJ* (maximum) and *psbF.* The observation that these initiation peaks are usually restricted to the first five to seven codons of an ORF raised the hypothesis that RNA hairpin structures downstream of the start codon could hinder the translocation of the ribosome after the first peptide bond is formed, leading to a longer dwell time on the start codon and therefore increasing the number of RPFs at this

position. To test this hypothesis, the minimum free energy (mfe) of every transcript's first 50 bases following the seventh codon was calculated as an estimate of the transcripts propensity to form RNA secondary structures within this region. However, comparing the mfe values with the ratio of average coverages (Figure 3-18 A), showed no correlation (Spearman's r -0.06), indicating that the reason for the differing heights of initiation peaks must be more complex and could potentially point to a widespread pausing mechanism taking place between translation initiation and elongation that might regulate translation output.

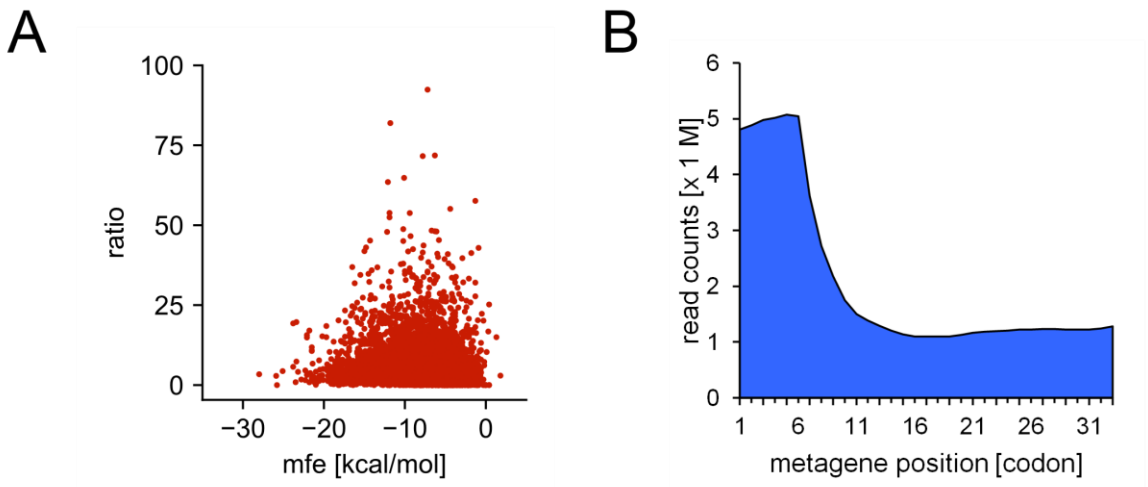


Figure 3-18: Features of initiation peaks. (A) Scatter plot opposing the minimum free energy in the 50 base-region downstream of the initiation region of all transcripts against the ratio of average coverage within their initiation region and the remaining ORF. (B) Metagene coverage plot representing the codon-wise sum of coverage of all annotated nuclear transcripts for the first 33 codons.

3.1.9 Analysis of ribosomal offsets provides deep insights into translational mechanics

Ribosomal P- or A-site offsets are the distance between an RPF's 5' or 3'-terminus and the location of the P-site of the ribosome that produced that footprint. With the help of these offsets, it is possible to calculate the P-site location of every RPF and determine the general frame-preference of the data set as a quality attribute (see section 3.1.2). However, analyzing the offsets on the metagene level in depth for all RPF species separately can also provide mechanistic insights about translation. P- and A-site offsets were determined by extracting all RPFs that covered an annotated start or stop codon and mapping the position of their 3' or 5'-end in respect to the first or last base of the start or stop codon they cover, respectively. Offset frequencies were counted for every RPF length species separately and depicted as a heatmap in Figure 3-19 A for the cytosolic RPFs. Interestingly, a prominent periodic pattern of 5'-offset frequencies can be observed for the main RPF species (30 nt) from 12 nt (large asterisk) downstream with very high offset counts occurring every 3 nt (small asterisks) (see Supplementary Figure 7 for detailed view). These reflect the codon-wise movement of the early elongating ribosomes and therefore the periodicity of the RPFs produced by ribosomes that already entered elongation but still covered a start codon. The most frequently observed 5'-offset of an RPF species was declared as the true offset for this length. For the main RPF species this offset was 12 nt (large asterisk) which is in good agreement with the 5'-P-site offset measured in yeast (Ingolia et al., 2009). It can be observed that the main 5'-offset within every single codon downstream of the main offset scatters diagonally from bottom left to top right when following the RPF length axis from long to short. This may potentially reflect elongation factors binding to the ribosome as well as structural rearrangements during the different stages of an elongation cycle, leading to a sometimes smaller or larger footprint of the ribosome on the transcript molecule. Considering that each RPF species has a distinct true 5'-P-site offset, the occurrence of the periodic pattern within the 5'-offset range of 12 nt to 27 nt in the 30 nt RPF species seems counterintuitive. The triplet-wise movement is a feature of elongation, but the ribosomes producing the RPFs in question could not have reached a start codon with their P-sites yet. Although eukaryotic cytosolic ribosomes are known to have a 5'-scanning mechanism for start codon detection this mechanism is known to work via pre-initiation complexes sliding randomly up- and down the 5'-UTR, not in a triplet-wise movement (Hinnebusch, 2014). It is more likely that the periodic traces of the ribosomes seen upstream of the main offset mainly originate from falsely annotated transcripts whose actual start codons are encoded upstream or from alternative transcript variants with an upstream start codon. The codon-wise movement of ribosomes is visible in the 3'-A-site offsets as well. In theory, the A-

and P-site offset together with the 6 nucleotides occupied by P- and A-site themselves should sum up to the total length of the RPF species in question. However, the 3'-A-site offset determined for the 30 nt species is only 9 nt which sums up with the 12 nt 5'-P-site offset and the 6 nt of A- and P-site themselves to only 27 nt in total. The missing three nucleotides can be explained by the translation termination mechanism, which starts when the stop codon is

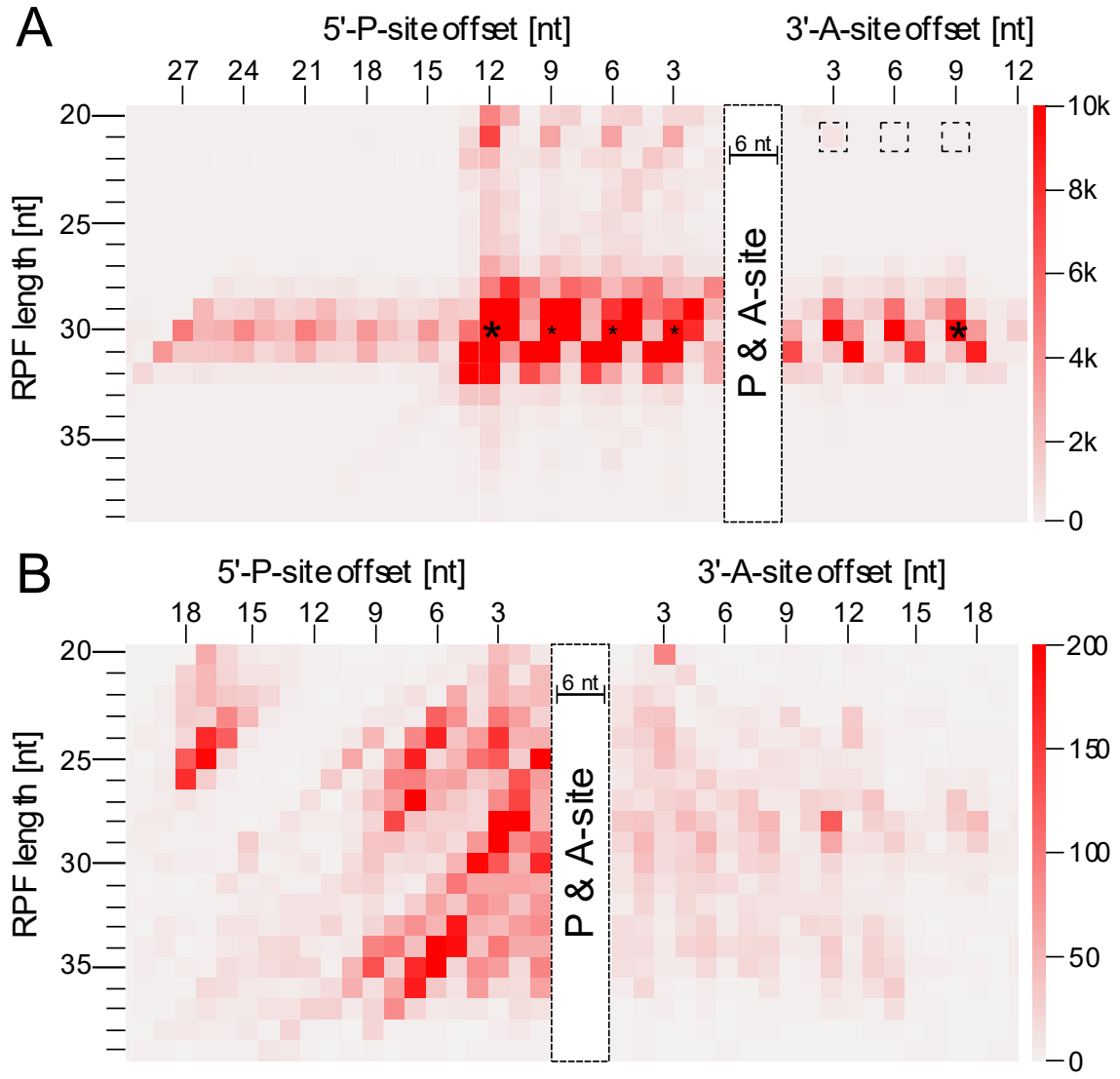


Figure 3-19: Ribosomal 5'-P-site and 3'-A-site offsets separately for RPF length species. (A) Heatmap representing the ribosomal offsets of cytosolic ribosomes. Small asterisks mark triplet-wise maximum accumulations of 5'-offset counts. Large asterisks mark global maximum accumulations of 5' or 3'-offset counts. Positions in dashed frames in the 21 nt RPF row represent the 3'-offsets that were observed in the 30 nt species to accumulate triplet-wise maximum counts (B) Heatmap representing the ribosomal offsets of chloroplast ribosomes. Offsets are represented by columns, RPF length species by rows. The color depth of the tiles represents the counts of RPFs per RPFs length species and offset. Modified from Gotsmann et al., 2024.

incorporated into the A-site of the ribosome and the last amino acid of the ORF is transferred onto the polypeptide chain (Hellen, 2018). Cycloheximide inhibits elongation by blocking the binding of aminoacyl-tRNA to the A-site prior to peptide bond formation (Schneider-Poetsch et al., 2010). When a ribosome after decoding the last codon of the ORF translocates and a stop codon is bound to the A-site, the ribosome is dissociated by a ribosome release factor into its

subunits and cannot be stabilized by cycloheximide. For this reason, most terminating ribosomes with the true 3'-A-site offset of 12 nt may be lost during sampling while the ribosomes decoding the stop-1 codon are stabilized by cycloheximide. Since the true offset of a species is normally the most frequently observed offset and since the stop-1 decoding ribosomes greatly outnumber the true terminating ribosomes, their offset can easily be mistaken as the true offset. This explanation is supported by the metagene P-site coverage profile (Figure 3-20 A) clearly showing a pronounced loss of observed P-sites from metagene codon 58 (stop-2, when the stop-1 codon is located at the A-site) to 59 (stop-1, when the stop codon is incorporated into the A-site, triggering termination) and as expected, only few P-sites covering the stop codon itself due to the dissociation of ribosomes after stop codon binding. It is important to check for this phenomenon and correct the observed 3'-A-site offset when it is used for determination of the P-site location, a done for chloroplast RPFs in this study.

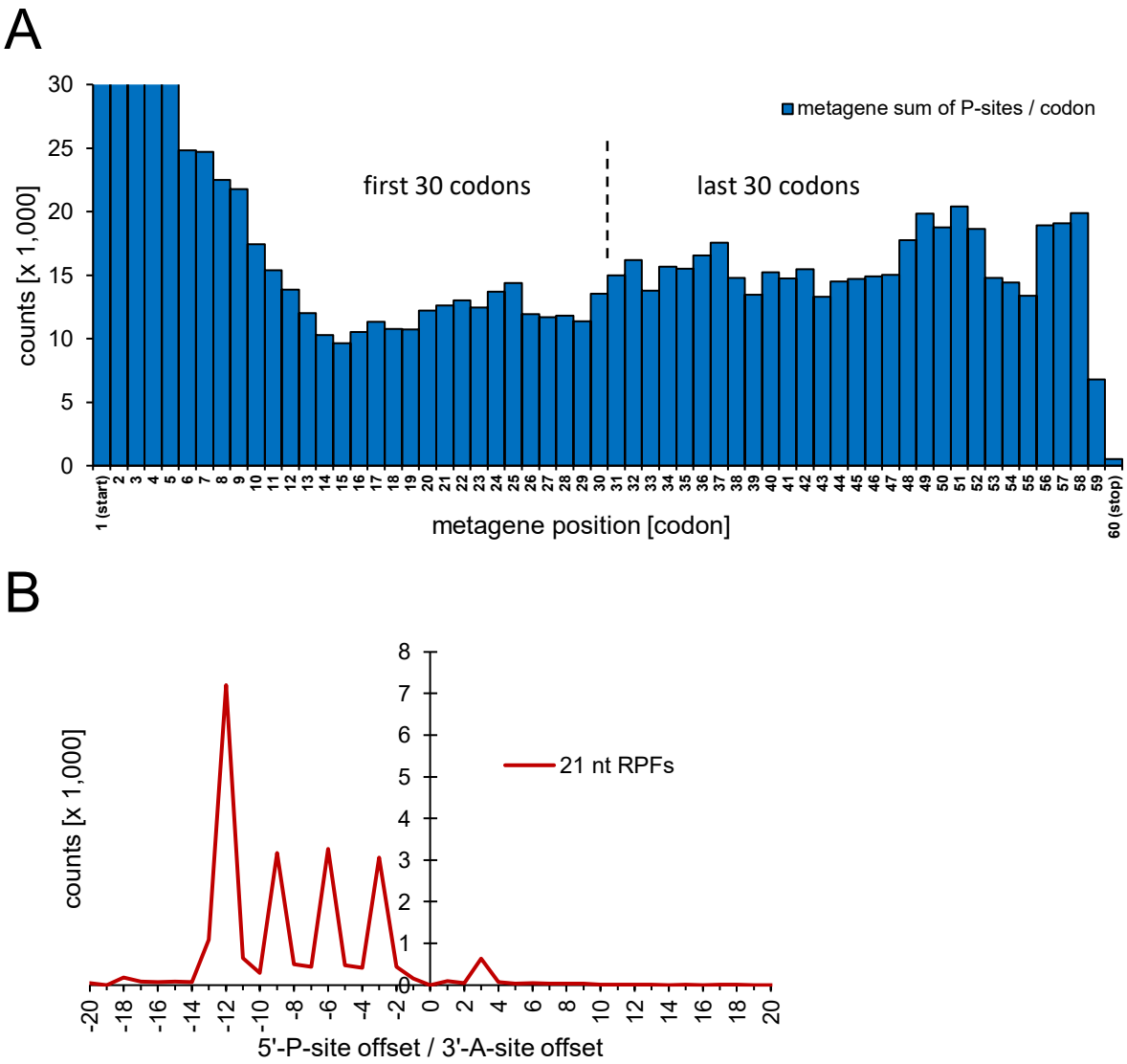


Figure 3-20: Details of ribosomal offsets. (A) Scaled metagene P-site profile displaying the sum of ribosomal P-sites counted on every codon for the first and last 30 codons, respectively. Calculations were performed on the 2,000 top-translated transcripts (including alternative transcripts). Columns of codons 1 to 5 were cut off for better visibility of the lower codon counts (1 – 5 counts: 259003, 136019, 97149, 67107, 43987). (B) Detailed distribution of 5'-P-site / 3'-A-site offsets of cytosolic 21 nt RPF species.

Interestingly, the 21 nt RPF species (called short species herafter) shows a periodic pattern among the 5'-P-site offsets very similar to the 30 nt species. Although this species is considerably shorter than the main RPF species, the pattern is formed by the same 5'-offsets, indicating that the short species results from degradation of the RPF exclusively 3' of the P-site. This is supported by their main 3'-A-site offset of 3 nt (Figure 3-20 B) for better visibility) which together with the 6 nt of P- and A-site and the main 5'-offset of 12 nt sums up to 21 nt. The difference between the short species' 3'-A-site offset and the corrected 3'-A-site offset (12 nt) of the main RPF species is 9 nt, which is also the length difference of both species, once more pointing towards degradation exclusively 3' of the P-site. Such a degradation could be effect of a conformational change 3' to the decoding center, exposing nine additional bases of the RPF to nucleic digest by the RNase. Such conformational changes are known to occur in transition states during the translocation from one codon to another, when the subunits of the ribosome are rotated against each other and during the decoding process (Brilot et al., 2013; Ling & Ermolenko, 2015; Salsi et al., 2014). Of note, different RPF lengths have been shown before to be related to different ribosomal conformations before (Lareau et al., 2014). Unlike the main RPF species' 3'-A-site offset, the short RPF species' 3'-A-site offset does not exhibit a discrepancy between the RPF length and the sum of all RPF parts (5'-A-site offset, A-site, P-site and 3'-A-site offset) and thus does not need correction. This indicates that this offset originates from ribosomes positioned with their A-sites on stop codons but that have not dissociated yet. Together with the inability to observe the true 3'-A-site offset within the main RPF species, this suggests that the ribosomal conformation that yields the short RPF species is accommodated before the stop codon is locked in the A-site and termination starts. Therefore, it can be considered a post-translocation conformation. Another explanation could be that the short species is a product of ribosomes with empty A-sites. During elongation, factors eEF1A and eEF2 and during termination the eRF1-eRF3 complex bind close to the A-site (Dever et al., 2018; Hellen, 2018) (Figure 1-4). It is possible that their presence may shield bases of the RPF 3' of the A-site from digest or expose them in their absence, respectively.

In contrast to the nuclear RPF offsets, the offsets of chloroplast RPFs do not expose a clear pattern. Among the 5'-offsets, multiple clusters are visible on the heatmap (Figure 3-19 B) which display neither a dominant offset nor a dominant RPF species. Among the 3'-A-site offsets, however, at least for the main RPF species of 28 nt length a slightly periodic pattern and a main offset of 11 nt can be observed. Considering that chloramphenicol does not inhibit termination either, the real 3'-A-site offset for the main species can be assumed to be 14 nt, resulting in a 5'-P-site offset of 8 nt. This matches a minor accumulation of 5'-termini on the heatmap. Due to the difficult assessment of the real 5'-P-site offsets among chloroplast RPFs, the P-site determination of chloroplast RPFs was performed using the 3'-A-site offsets, which appeared less problematic. The unclear distribution of 5'-P-site offsets presumably is a

consequence of the trimodal RPF length distribution in the chloroplast (see section 3.1.5), which especially due to the overlapping RPF length regions between the three peaks may prevent a precise offset determination.

3.2 Ribo-Seq of a uS11c depleted mutant strain

3.2.1 Differentially translated transcripts reflect morphological changes of the S11kd strain in response to uS11c depletion

To demonstrate the capabilities of the Ribo-Seq method for the detection of differentially translated transcripts under varying conditions, a Ribo-Seq experiment was conducted with an inducible knock down strain of the chloroplast ribosomal protein uS11c. In an earlier study using a microarray-based ribosome profiling approach, uS11c was found to be differentially translated between mixotrophic and photoautotrophic growth conditions in *C. reinhardtii* (Trösch et al., 2018) with its expression being reduced under photoautotrophic conditions. Intuitively, the differential expression of a ribosomal protein should disturb the stoichiometry of ribosomal proteins and cause severe defects. Thus, it was hypothesized that uS11c could represent a translational modulator rather than an essential component of the chloroplast ribosome. uS11c is located at the 30S subunit platform region, close to the ribosomal E-site at the mRNA exit channel and the Shine-Dalgarno cleft, the region that harbors the anti-Shine-Dalgarno sequence and facilitates translation initiation of Shine-Dalgarno sequence containing transcripts in *E. coli* (Figure 3-21 A). Furthermore, its C-terminus is in close contact to the mRNA and together with ribosomal protein uS7c it forms a 'latch' that threads the 5'-end of the translated mRNA through the Shine-Dalgarno cleft upon translocation of the ribosome. The potential regulatory relevance of uS11c's contact sites support the hypothesis that it plays a role in regulating chloroplast translation. In a follow-up study (Gotsmann, 2018), inducible knock down strains of uS11c were generated (S11kd) by coupling the *rps11* coding sequence to the *psbD* 5'-UTR, facilitating translational control of the uS11c protein to the nuclear encoded T-factor NAC2. In the background strain used for this mutant (Ramundo et al., 2013), the NAC2 expression in turn is coupled to the thiamin-responsive *THI4*-riboswitch and the vitamin B12-responsive *METE* promoter that both suppress expression of the NAC2 protein upon addition of their respective responsive agent. To uncouple D2 expression from NAC2, the *psbD* 5'-UTR was exchanged against the *psaA* 5'-UTR in the background strain. Under mixotrophic conditions, the knock down mutants reached high levels of uS11c depletion (~90%) and proved to be viable even after prolonged periods of uS11c depletion. However, drastic morphological changes like the pronounced formation of palmelloid colonies, fragmentation of the chloroplast, loss of chlorophyll and increased formation of starch granules were observed upon induction of the knock-down (Figure 3-21 C and D). It was found that uS11c seems to be essential for expression of photosystem II reaction center proteins PsbA and PsbD but its depletion had no effect on protein levels of the ATPase subunit AtpB. Regarding ribosome associated proteins, the induced mutants showed no difference in accumulation of the nuclear encoded chloroplast

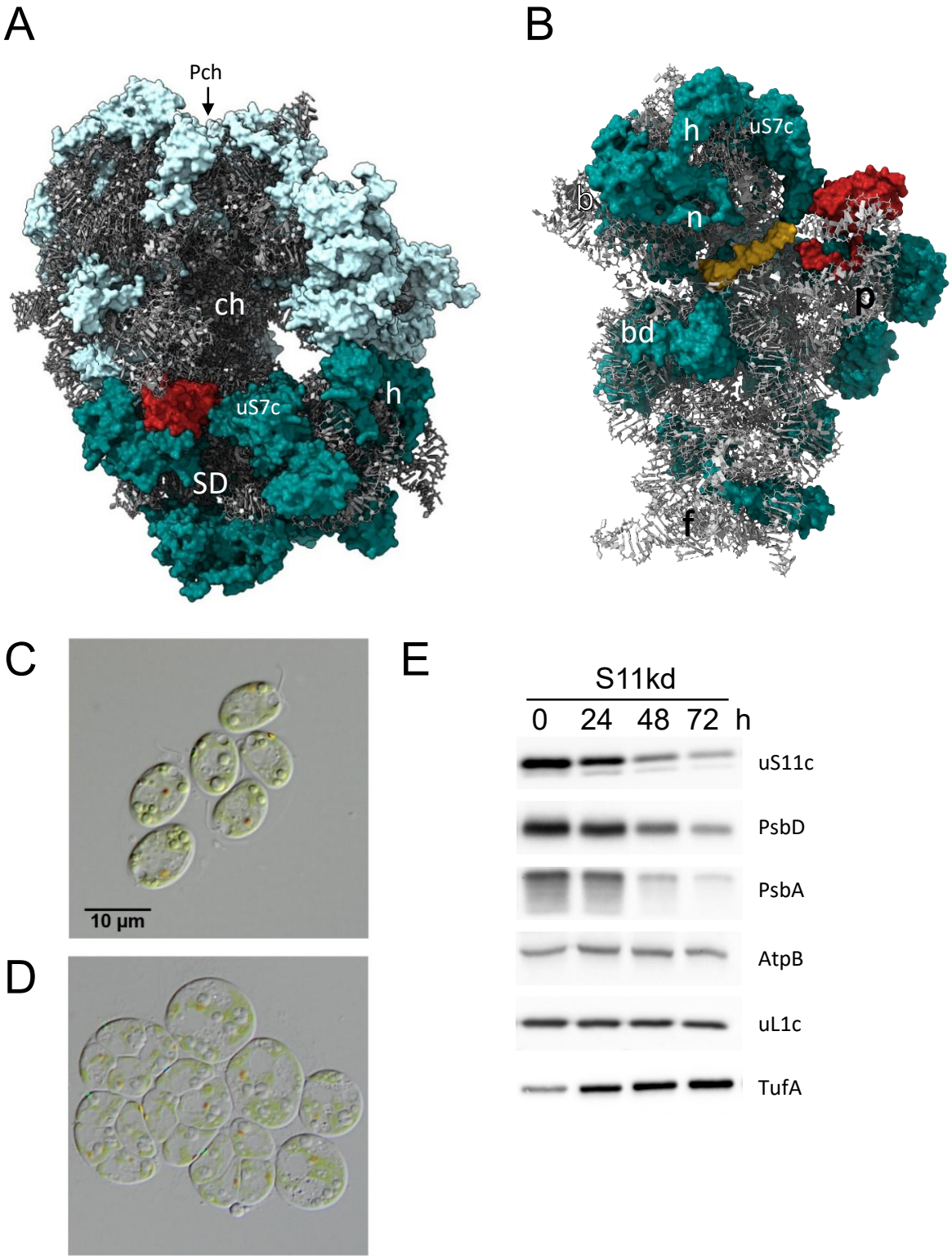


Figure 3-21: Details of chloroplast ribosomal protein uS11c. (A) View on the E-site of the chloroplast ribosomes mRNA exit channel. rRNA is depicted as grey helices, proteins of the 50S subunit are depicted in light blue, proteins of the 30S subunit are depicted in teal. uS11c is depicted in red. (ch) mRNA exit channel (uS7c) chloroplast ribosomal protein S7 (SD) Shine-Dalgarno cleft (h) 30S subunit head reagion (Pch) approximate position of the polypeptide exit channel. (B) View on the subunit interface of the chloroplast 30S ribosomal subunits. Coloring according to A. Orange surface depicts mRNA bound to the ribosome. (h) head region (b) beak region (n) neck region (p) platform (f) foot region (bd) body (uS7c) chloroplast ribosomal protein S7. (C) Morphology of S11kd background strain (A31) after 96 hours of *NAC2* suppression. (D) Morphology of S11kd strain after 96 hours of *NAC2-rps11* suppression. (E) Immunoblots of S11kd after 0, 24, 48 and 72 hours of *NAC2-rps11* suppression. (uS11c) chloroplast ribosomal protein S11 (PsbD) photosystem II reaction center protein D2 (PsbA) phosotsystem II reaction center protein D1 (AtpB) chloroplast ATPase subunit CF1-β (uL1c) nuclear encoded chloroplast ribosomal protein L1 (TufA) chloroplast translation elongation factor EF-Tu. (A and B) modified from Bieri et al., 2017 PDB ID 5mmm. (C to E) from Gotsmann, 2018 (master thesis).

ribosomal protein uL1c, indicating that ribosome biogenesis was not affected by the depletion of uS11c. Surprisingly, levels of the chloroplast encoded translation elongation factor TufA (homolog of bacterial EF-Tu) even increased dramatically, suggesting that uS11c is indeed dispensable for effective translation of certain chloroplast encoded proteins (Figure 3-21 E).

The Ribo-Seq experiment was conducted with biological duplicates of two independent S11kd mutants (S11kd2 and S11kd5) and samples were harvested under mixotrophic conditions (50 µE continuous illumination, 25°C) before induction (ctrl) and 48 h after induction (induced). The sampling time point after induction was chosen as a compromise of depletion efficiency and time to avoid secondary effects triggered by depletion of uS11c-affected proteins as far as possible. Since the experiment was conducted before validation of Ribo-Seq protocol iii, the samples were processed according to protocol i. Moreover, the Ribo-Seq experiment was complemented with an RNA-Seq experiment from the same cultures. Quality control of the Ribo-Seq samples (see Supplementary Figure 8 and Supplementary Figure 9) displayed an in-frame preference of approximately 50% for cytosolic ribosomes in all samples and an RPF length distribution of cytosolic RPFs ranging from 30 to 35 nt with clear maxima at 32 or 33 nt. Compared with the distribution of protocol i in Figure 3-2 A, this is slightly larger. However, it can be assumed that the nuclease digest is also subject to experiment-specific variations and may yield slightly different outcomes in different experiments. The RPF length distribution of chloroplast encoded transcripts also differed slightly from its counterpart in Figure 3-2 A in that the distinct modes of the length distribution were less pronounced, also pointing to a less effective nuclease digest of the samples. Calculation of frame preference for chloroplast encoded transcripts failed due to a shortage of RPFs mapping to start or stop codons which caused offset determination to fail due to too many ambiguities. However, according to Figure 3-2 C, there was no clear frame preference expectable with the applied protocol. The biotype distribution of both cytosolic and chloroplast RPFs showed a clear dominance of CDS-mapping RPFs, ensuring the data sets represent RPFs. After quality control was completed, reads for all main transcripts were counted using Ribo-Seq functions and then analyzed for differentially expressed transcripts using the DESeq2 package.

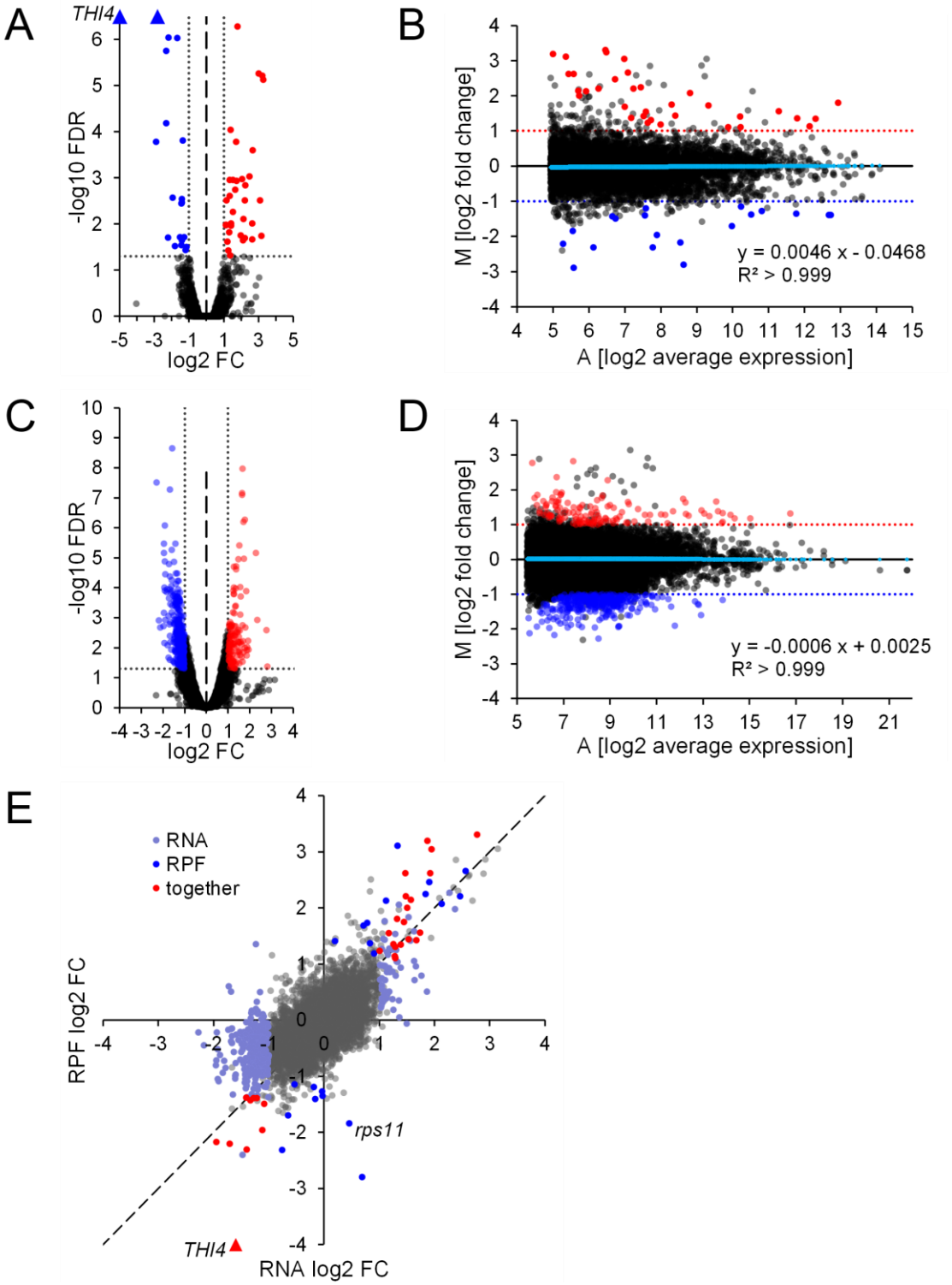


Figure 3-22: Differential expression analysis of uS11c depleted strains. (A) Volcano plot representing transcripts in Ribo-Seq experiment by Benjamini-Hochberg corrected p-values (-log10 FDR) versus log2 fold change (log2 FC) of ctrl / induced samples. Black dots represent transcripts not differentially expressed on the translational level, blue and red dots represent translationally up- and downregulated transcripts, respectively. Dashed vertical line indicates 0 change, dotted grey lines represent borders of significance on both axes. Blue triangles represent transcripts *THI4* and *Cre13.g573900* which are out of range of the plot. (B) MA-plot representing transcripts in Ribo-Seq experiment by their log2 average expression between both conditions and their log2 fold change. Colors like A. Blue and red dotted lines represent the borders of significance on the log2-fold change axis. Cyan dots represent the LOWESS curve fitted into the scatter cloud and y and R^2 represent the formula of the linear regression line based on the LOWESS curve and its coefficient of determination. (C and D) Like A and B for RNA-Seq experiment. (E) log2 fold change on translational level (RPF log2 FC) versus transcriptional level (RNA log2 FC). Grey dots represent transcripts not differentially expressed on any level, purple dots represent transcripts differentially expressed (DE) on transcript level, blue dots represent transcripts DE on translational level and red dots represent transcripts DE on both levels. Borders of significance: FDR < 0.05 and log2 FC > 1.

On the translational level, in total 19 transcripts were significantly downregulated in response to uS11c depletion and 33 transcripts were significantly upregulated. On transcript level instead, 341 transcripts were significantly down- and 126 were upregulated (Figure 3-22 A and C). For both experiments, the MA plots (Figure 3-22 B and D) show that the scatter cloud aligns with the 0-change line. A LOWESS curve (Cleveland, 1981) was calculated for both scatter clouds (cyan dots) to determine if any of both conditions yield higher change values, which would require smoothing of the cloud. Linear regression lines were fitted to the LOWESS curves and indicated an almost flat line at log2 FC 0 with only minimal slope and offset in both cases, suggesting the data sets were free of technical bias and therefore no smoothing of the data was necessary. To compare both experiments with each other, the changes on both levels were plotted against each other (Figure 3-22 E). Interestingly, although a much greater number of transcripts was differentially expressed on transcript level than on translational level, these do not include all the translationally differentially expressed (DE) transcripts. Instead, of the 19 downregulated candidates, nine were non-DE on the transcript level (Table 3-2, bottom). Among the 33 translationally upregulated transcripts on the other hand, 12 were classified as non-DE on transcript level. Of note, no candidates with opposing directions of regulation between translation and transcript level were found, as can be seen by the absence of red data points in the upper left or the lower right panel of Figure 3-22 E. The translationally most downregulated transcript was *THI4* (-6.15 log2 FC – resembling ~1.4% residual expression), encoding the thiazole biosynthetic enzyme that harbors the riboswitch in its 5'-UTR that was also fused to the *NAC2* transcript in the mutant, exemplifying the efficacy of the riboswitch. *rps11* was downregulated by -1.84 log2 FC (resembling ~28% residual expression) proving that the knock down was successfully induced, inhibiting translation of uS11c. However, *NAC2* surprisingly was found to be neither translationally nor transcriptionally downregulated. On both levels, adjusted p-value and log2 fold change (translational ~0.4, transcriptional ~-0.05) were insufficient to suggest a true change. Considering the low expression values of *NAC2* (RPKM values range from 3.15 to 5.46 in the Ribo-Seq experiment and from 3.85 to 5.26 in the RNA-Seq experiment), it is likely that the number of replicates and sequencing depth were both too low to reliably detect a differential expression of the transcript. Despite their clear decrease on the protein level upon uS11c depletion, neither *psbA* nor *psbD* was found to be downregulated in any of both data sets. Instead, photosystem II subunits *psbC*, *ycf12* and *psbK* were found to be downregulated on translational but not on transcript level (Table 3-2, lower part), indicating that their change is attributable to the depletion of uS11c and not to transcriptional changes. The transcriptional changes of these candidates, albeit not significant anyway, range in a very narrow window close to 0 (-0.03 to -0.18). These results indicate that PsbA and PsbD proteins may be synthesized normally in absence of uS11c but are possibly degraded afterwards due to the lack of downstream assembly partners, explaining their decline

on protein level. Apart from *atpH*, all remaining candidates are cytosolic translated transcripts whose translation is independent from uS11c and therefore must be regulated in response to the added vitamins or the changes in the chloroplast. These include *THIC1* and *Cre13.g573900*. While *THIC1* is known to be involved in thiamin metabolism, the other is only proposed to be a Na^+-thiamin symporter. Regarding the excess of extracellular thiamin added upon knock-down induction, it would make sense to limit thiamin uptake by regulating the expression of such a transporter. Furthermore, transcripts encoding both alpha tubulin isoforms, TUA1 and 2 are downregulated on the transcriptional and translational level, which might contribute to the morphological changes of the knock down strains seen in Figure 3-21 D. Among the translationally upregulated transcripts (Table 3-3), *tufA* is the strongest upregulated chloroplast encoded candidate (log2 FC 1.8, resembling approximately 3.5-fold expression), clearly reflecting the strong increase seen on protein level (Figure 3-21 E). Besides *tufA*, the only chloroplast encoded candidates are *rpl5* and *rps2*, both encoding ribosomal proteins. Possibly, the upregulation of ribosomal constituents may represent a rescue mechanism when translational defects are sensed in the chloroplast. A striking detail of the list of upregulated transcripts is the dominance of transcripts encoding proteins targeted to the secretory pathway (20 of 33).

Table 3-2: Transcripts downregulated on the translational level in response to uS11c knock-down induction. The table is split into transcripts downregulated on the translational and transcript level (top) and transcripts only downregulated on the translational level (bottom). For every transcript, the log2 fold change (log2 FC) on both levels is indicated as well as the localization of the encoded protein according to experimental evidence or PredAlgo prediction. If the function of a protein is only inferred based on homology, it is given in parentheses. Chloroplast-encoded candidates are indicated by red font.

RNA + RPF significantly downregulated				
gene	**RNA log2 FC**	**RPF log2 FC**	**(predicted) localization**	**(proposed) function**
THI4	-1.59	-6.15	chloroplast	thiamine metabolism
NFO2	-1.40	-2.30	chloroplast	flavin metabolism
Cre02.g096000	-1.71	-2.21	cytosol	(sulfate transporting ATPase)
Cre02.g095080	-1.95	-2.18	secretory pathway	(major vault protein)
Cre12.g489300	-1.12	-1.96	cytosol	unknown
FAP344	-1.08	-1.49	cytosol	flagellar associated
Cre01.g013500	-1.34	-1.43	cytosol	unknown
TUA1	-1.28	-1.39	cytosol	cytoskeleton
TUA2	-1.21	-1.39	cytosol	cytoskeleton
HSP90A	-1.41	-1.38	cytosol	chaperone
RPF significantly downregulated				
Cre13.g573900	0.70	-2.80	secretory pathway	(Na^+-solute symporter, thiamin)
Cre17.g721450	-0.76	-2.31	secretory pathway	(Na^+-solute symporter, urea)
rps11	0.46	-1.84	chloroplast	chloroplast ribosomal protein
THIC1	-0.65	-1.70	chloroplast	thiamine metabolism
ycf12	-0.16	-1.41	chloroplast	PS II
psbC	-0.03	-1.35	chloroplast	PS II
atpH	-0.04	-1.27	chloroplast	ATPase
psbK	-0.18	-1.20	chloroplast	PS II
RYR1	-0.53	-1.15	cytosol	Ca^{2+} channel

A large portion of these candidates (*PHC6, PHC19, PHC61, PHC12, PHC22, PHC4*, *CWP1 CWP2, VSP1*) encode proteins annotated as cell wall components and may be associated with the formation of pamelloid colonies observed upon uS11c depletion. These are accompanied by *GOX6* and *GOX19*, putative glyoxal or galactose oxidases. Interestingly, glyoxal oxidases were found to be associated with extracellular carbohydrate degradation in wood degrading fungi (Kersten & Kirk, 1987). Potentially this class of enzymes serves a similar role in *C. reinhardtii* for remodeling of the glycoprotein-rich cell wall (Goodenough & Heuser, 1985; Poulhazan et al., 2024). Another potential cell wall remodeling enzyme might be MMP33, a putative matrix metallo protease that is also translationally upregulated upon uS11c depletion.

Table 3-3: Transcripts upregulated on the translational level in response to uS11c knock-down induction. The table is split into transcripts downregulated on the translational and transcript level (top) and transcripts only downregulated on the translational level (bottom). For every transcript, the log2 fold change (log2 FC) on both levels is indicated as well as the localization of the encoded protein according to experimental evidence or PredAlgo prediction. If the function of a protein is only proposed based on homology, it is given in parentheses. Chloroplast-encoded candidates are indicated by red font.

RNA + RPF significantly upregulated				
gene	**RNA log2 FC**	**RPF log2 FC**	**(predicted) localization**	**(proposed) function**
Cre09.g397623	2.77	3.30	secretory pathway	(extracellular protein kinase)
GOX6	1.87	3.19	secretory pathway	(glyoxal / galactose oxidase)
PHC6	1.94	3.04	secretory pathway	cell wall component
PHC19	1.92	2.62	secretory pathway	cell wall component
PHC61	1.47	2.62	secretory pathway	cell wall component
GOX19	1.48	2.21	secretory pathway	(glyoxal / galactose oxidase)
Cre14.g618550	1.57	2.14	secretory pathway	unknown
Cre03.g204450	1.50	2.00	secretory pathway	unknown
tufA	1.32	1.80	chloroplast	chloroplast translation elongation
PHC12	1.45	1.74	secretory pathway	cell wall component
CWP2	1.74	1.56	secretory pathway	cell wall component
rpl5	1.18	1.55	chloroplast	chloroplast ribosomal protein
FAP211	1.53	1.44	secretory pathway	flagellar associated protein
Cre06.g261450	1.68	1.43	cytosol	(high mobility group protein)
FAP102	1.26	1.35	secretory pathway	flagellar associated protein
FEA1	1.39	1.34	secretory pathway	iron uptake
LCYE1	1.29	1.30	chloroplast	carotenoid synthesis
Cre04.g217934	1.00	1.23	cytosol	unknown
FOX1	1.28	1.14	cytosol	iron uptake
CMP1	1.29	1.11	chloroplast	(carbamoyl phosphate synthase)
FER1	1.29	1.11	chloroplast	iron storage
RPF significantly upregulated				
MMP33	1.33	3.11	secretory pathway	(matrix metallo protease)
PHC22	2.57	2.65	secretory pathway	cell wall component
ARS12	1.90	2.46	secretory pathway	aryl sulfatase
PHC4	1.84	2.24	secretory pathway	cell wall component
VSP1	2.46	2.21	secretory pathway	cell wall component
CDP1	1.12	2.12	secretory pathway	ammonia signalling
CWP1	2.13	2.07	secretory pathway	cell wall component
PTA3	0.79	1.73	cytosol	(H^+/Phosphate symporter)
NAGS1	0.72	1.68	chloroplast	N-acetylglutamate synthase
rps2	0.20	1.41	chloroplast	chloroplast ribosomal protein
Cre07.g800817	0.84	1.37	chloroplast	RNA binding
Cre09.g801014	0.91	1.19	cytosol	unknown

However, apart from transcripts encoding cell wall associated proteins, the list also contains candidates annotated as involved in iron uptake and storage (*FEA1, FOX1, FER1)* and several others. An especially interesting candidate is the uncharacterized transcript *Cre06.g261450*, encoding a putative “high mobility group protein”. While these proteins usually serve a function in chromatin organization in the nucleus, they were found to be secreted by mammalian cells in response to stress stimuli and act as extracellular signaling protein (Yang et al., 2010). It may be speculated that this function is conserved in *Cre06.g261450*.

In summary, the differential expression analysis yielded only four candidate transcripts whose translation may be strictly dependent on uS11c (*ycf12, psbK, atpH* and *psbC*). While the loss of PsbA and PsbD protein during uS11c depletion cannot be attributed to a lower translational activity, the translational downregulation of the three photosystem II subunits *ycf12*, *psbK* and *psbC* might cause severe problems in the photosystem II assembly pathway that could finally lead to degradation of uncomplexed PsbA and PsbD protein. The increase of TufA protein during uS11c depletion instead can be attributed to a clear increase of transcript abundance and translational activity of the transcript. Apart from direct consequences of uS11c depletion on chloroplast translation, a selection of transcript candidates was identified that may cause the severe morphological phenotype of the knock down. A striking observation of these experiments is the large difference between the number of translationally differently expressed transcripts and the number of transcriptionally differently expressed transcripts. This difference might indicate that the Ribo-Seq experiment suffers from low number of replicates, especially regarding the detection of DE transcripts with rather low amplitude of change. However, from a biological point of view, this observation could also underline that transcriptional changes may not necessarily serve as proxy for proteomic changes but rather underly buffering mechanisms that fine tune the expression of a transcript on translational level.

3.2.2 Inspection of ribosome profiles reveals changes in translation dynamics of some transcripts affected by uS11c depletion

To gain a deeper understanding of the effects of uS11c depletion on chloroplast translation, the ribosome profiles of negatively affected chloroplast transcripts were normalized by calculating the percentage share of every nucleotide on the total coverage of a transcript. Afterwards, profiles were averaged for controls and induced samples, respectively and compared against each other. Figure 3-23 displays the coding sequence regions of *rps11*, *psbK*, *atpH* and *ycf12* (see Supplementary Figure 11 for respective full profiles) and the full profile of *psbC*. The profile of *rps11* (Figure 3-23 A) itself in induced knock-down cultures is dominated by noise, especially in the range of codons 81 to 131. Due to the lower translation of rps11 after induction (average RPKM 15 in induced cultures versus average RPKM 49.5 in controls) the total read count of the transcript is low, leading to a larger variation of normalized coverage due to small differences between the replicates. While the profiles of *psbK* and *ycf12* (Figure 3-23 B and C) only show small deviations from their reference profiles, the profiles of *atpH* and *psbC* (Figure 3-23 D and E) show considerable differences compared to their reference profiles. In case of *atpH*, the reference profile displays a pronounced peak approximately covering codons 5 to 15, which is greatly diminished in the induced samples. Instead, coverage in the 5'-UTR directly upstream of the start codon and of the first five codons is slightly higher, albeit also having greater variation. Considering that uS11c is located closely to the Shine-Dalgarno cleft of the ribosomal 30S subunit, a region highly important for Shine-Dalgarno dependent translation initiation in bacteria, this finding may suggest that initiation of *atpH* is mechanistically disturbed by the depletion of uS11c, leading to a slower initiation and therefore accumulation of RPFs mapping to the respective sequence of the transcript. This hypothesis is further supported by the finding that the 5'-UTR of *atpH* contains a perfect Shine-Dalgarno sequence (AGGAGG, perfectly complementary to the ribosome's anti-Shine-Dalgarno sequence CCUCCU) only eight nucleotides upstream of the start codon (5'-**AUUUAGGAGGAAAUACAAUG**-3') (Shine & Dalgarno, 1974). Of note, the 5'-UTR of *psbK* also harbors an imperfect Shine-Dalgarno-like sequence seven nucleotides upstream of the start codons (5'-**UAAUGAAGGAUAUUUAUG**-3'). The profile of *atpH* shows a second considerable deviation from the reference profile in the region around codon 41, where coverage is greatly increased. Considering the length of the ribosomal polypeptide exit tunnel which shields approximately 30 to 40 amino acids N-terminal to the decoded position of the ribosome (Chadani et al., 2021), this region is presumably decoded when the N-terminus of the protein is about to emerge from the ribosome. Since AtpH is an integral membrane protein (Hahn et al., 2018), its translation may be accompanied by direct insertion into the thylakoid

membrane. Considering this, the accumulation of RPFs in this region may suggest an additional problem with co-translational membrane insertion in absence of uS11c. The profile of *psbC* (Figure 3-23 E), similarly to the profile of *psbK*, shows only small deviations from the reference profile, of which the most prominent one is a strong accumulation of RPFs located 5'-terminally of the 5'-UTR. Such an accumulation suggests very early ribosome binding to the transcript and might indicate a eukaryote-like start codon scanning mechanism (Hinnebusch, 2014) of chloroplast ribosomes for *psbC* that would be impaired by the depletion of uS11c. Interestingly, *psbC* harbors a regulatory *cis*-acting element within the middle of its 5'-UTR (nucleotides 222 – 320) that was shown to be essential to promote translation initiation (Rahim et al., 2016). Given the large distance of this element upstream to the start codon and assuming that a physical interaction between the ribosome and this element is necessary to initiate translation of *psbC*, it would make sense that ribosomes would bind upstream to the regulatory region and scan into 3'-direction. Moreover, the *psbC* transcript also contains an imperfect Shine-Dalgarno-like sequence ten nucleotides upstream of its non-canonical GUG start codon (5'-**UUUGUACGGAGGUAAUGCAAAGUG**-3') that was also identified to be required for translation (Rochaix et al., 1989; Zerges et al., 2003).

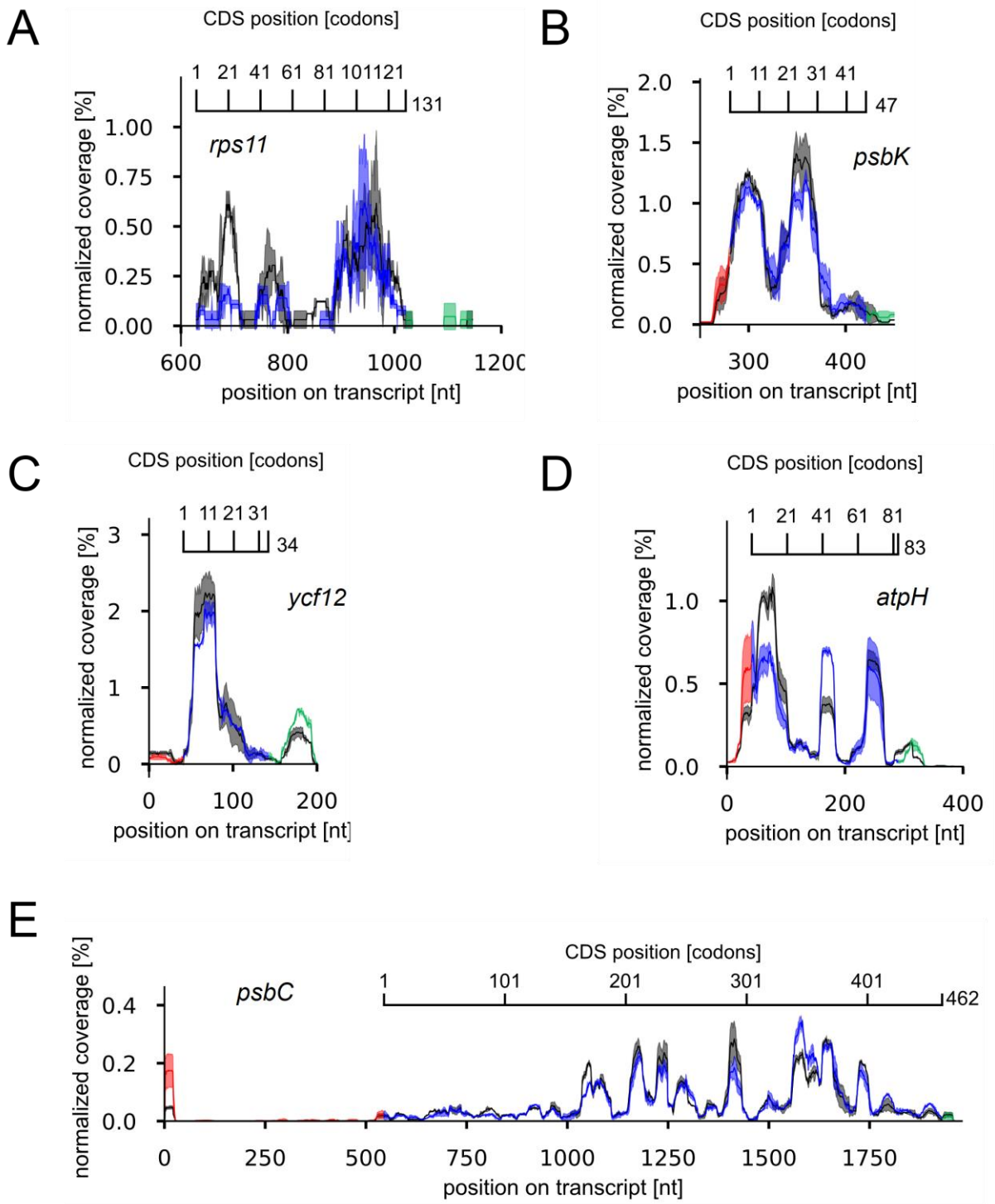


Figure 3-23: Comparative ribosome profiles of chloroplast transcripts negatively affected by uS11c depletion. (A – E) Transcript names are indicated in italic font. Black lines represent averaged reference profiles of control cultures. Colored lines represent the averaged profiles of induced cultures. Red parts represent 5'-UTRs, blue parts represent coding sequences and green parts represent 3'-UTRs. Shaded areas represent the standard deviation-based error bands. Scale bars on top indicate the codon-wise position along the coding sequence (CDS).

Although *psbA* and *psbD* were not found to be differentially expressed neither in the RNA-Seq nor the Ribo-Seq experiment, their protein levels were heavily affected by the depletion of uS11c. Their ribosome profiles after uS11c depletion display slight deviations from their

references (Figure 3-23 A and B). For *psbA*, these deviations are located within the 5'-UTR shortly upstream of the start codon and within the region around codon 145. In the first case, again a Shine-Dalgarno-like sequence is covered by RPFs over-accumulating compared to the reference profile (5'-**ACGGAGAAATTAAAACTTTAAAAAAATTAACATATG**-3'). However, due to the distance of 28 nucleotides between the sequence and the start codon, it is unlikely that a ribosome is able to bind the Shine-Dalgarno-like sequence while initiating translation with the start codon bound to its P-site, thus indicating that this region may play a role in ribosome binding but not in initiation itself. Interestingly, the second deviation from the reference profile is located at the same position that was earlier found to yield unusually short RPFs (see paragraph 3.1.5), indicating that whatever event is taking place at this position, it is affected by the absence of uS11c from the translating ribosome by displaying a considerably lower coverage at this site. The *psbD* profile shows deviations from the reference mainly locating to the second half of the coding sequence with the largest deviation found close to the stop codon approximately at codons 335 – 345. Although both transcripts were non-differentially expressed in the Ribo-Seq experiment, the deviations found in their profiles suggest that some mechanistic details of their translation process may change in absence of uS11c, nonetheless. Exemplarily comparing *psbA* and *psbD* to the profiles of *psaA* (Figure 3-24 D), another transcript non-DE in both experiments, shows that the differences seen in their profiles are indeed unusual and supports that they represent true effects of uS11c depletion rather than variance.

While *psbA* and *psbD* both were negatively affected on the protein level, *tufA* strongly accumulated during the knock-down, suggesting that uS11c is not required for translation of the elongation factor. The accumulation seen on protein level was verified with *tufA* being significantly upregulated both on transcriptional and translational level, making it another interesting candidate for closer investigation. The profile displays only slight deviations from the reference that may be attributable to biological variance rather than to uS11c depletion. However, a suspicious detail of the *tufA* ribosome profile is the absence of any coverage within the coding sequence up to codon 45. This finding does apply to both, the reference profile and the profile of the knock-down sample and subsequently was also verified in all other samples analyzed in this study. The respective sequence region was aligned to the whole genome of *C. reinhardtii* to investigate the possibility of a mapping error due to identical mapping positions but returned no match. This indicates that the absence of coverage within the first 44 codons is not a technical error. Gaps within a ribosome profile may occur due to low translation rate of a transcript or fast translation dynamics in part of a transcript. However, *tufA* is strongly expressed on the translational level (mean RPKM control: 735, mean RPKM induced: 2779) and gaps within the profiles of other transcripts within this range are usually are very small (see Figure 3-24 A, B, D), indicating that the gap within the *tufA* profile might be attributable to

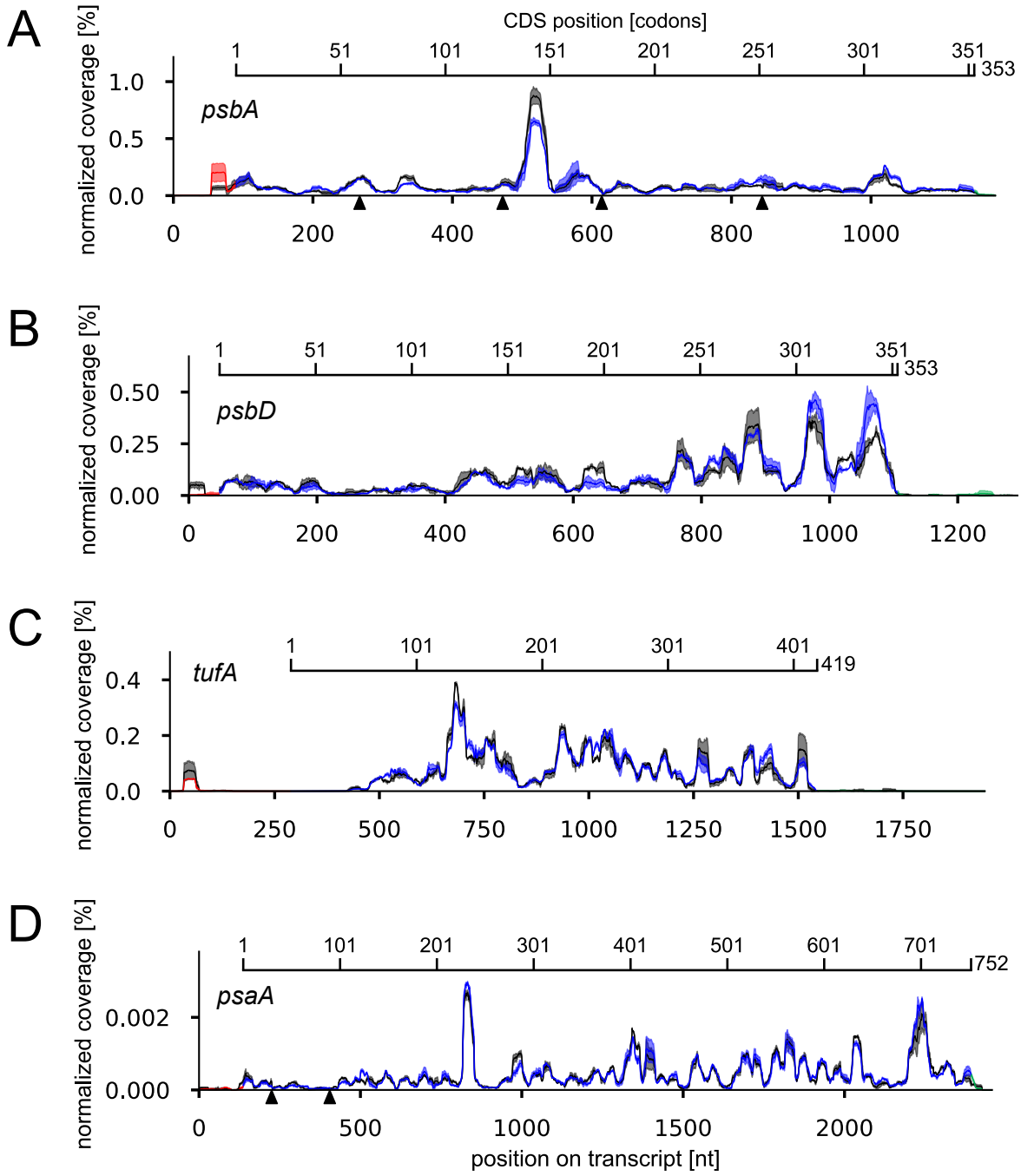


Figure 3-24: Comparative ribosome profiles of chloroplast transcripts. (A – D) Transcript names are indicated in italic font. Black lines represent normalized and averaged reference profiles of samples prior to knock-down induction. Colored lines represent the normalized and averaged profiles of induced cultures. Red parts represent 5'-UTRs, blue parts represent coding sequences and green parts represent 3'-UTRs. Shaded areas represent the standard deviation-based error bands. Scale bars on top indicate the codon-wise position along the coding sequence (CDS). Black arrows indicate exon-exon borders.

misannotation of the start codon. If that would be the case, alternative translation start sites within the close range of the onset of coverage in the coding sequence could be codons 50, 61 and 63, all encoding isoleucine with the near-cognate start codon "AUU". The closest alternative cognate start codon would be codon 92.

In summary, manual inspection of the ribosome profiles of uS11c-affected transcripts revealed that three of the four transcripts that were negatively affected on the translational level contained Shine-Dalgarno(-like) sequences close to their start codons (*atpH*, *psbK* and *psbC*). For *psbK* and *ycf12*, changes in the translation profiles were rather marginal, suggesting that their lower translation levels might be attributable purely to a lower propensity of initiation. For *atpH* instead, larger deviations from the reference profile were detected that may suggest problems with co-translational processes upon uS11c depletion. This hypothesis is supported by the finding that also the profiles of *psbA* and *psbD,* which were strongly reduced on protein level but not on translational level, showed considerable changes in some regions. *AtpH* also showed a deviation from the reference profile at its initiation site which contains a perfect Shine-Dalgarno sequence (while the other transcripts investigated only contain imperfect Shine-Dalgarno-like sequences). This might indicate that the absence of uS11c from a ribosome fully abolishes the ability to initiate transcripts depending on Shine-Dalgarno-like sequences for ribosome binding (*psbK* and *psbC*), which due to the incomplete depletion of uS11c in this experiment could explain their lower expression on the translational level but unchanged translation dynamics. The occurrence of a perfect Shine-Dalgarno sequence (like for *atpH*) instead might allow initiation even in absence of uS11c, due to stronger interactions between nucleotides of the transcript and the rRNA. Besides a generally diminished initiation rate, the absence of uS11c could change the dynamics of initiation in such a case as seen by changes within the initiation region of the profile. However, it must also be stated that the general relevance of Shine-Dalgarno-like sequences in chloroplasts is a subject of ongoing discussion and only a fraction of plastid encoded genes might be affected by the topic at all (Fargo et al., 1998; Scharff et al., 2017). At last, by inspection of the ribosome profile of *tufA*, which is translationally upregulated upon uS11c depletion, it was found that the annotation of the transcript might contain severe errors.

3.3 Selective Ribosome Profiling in *C. reinhardtii*

3.3.1 cpSRP54-specific Ribosome Profiling leads to global enrichment of footprints of nucleus-encoded transcripts

Apart from analyzing the translatome of *C. reinhardtii*, an additional aim of this study was to modify the Ribo-Seq protocol for the identification and analysis of target transcripts of specific translation co-factors. With the establishment of an in-lysate digest in protocol iii, the foundation for such a selective ribosome profiling (seRP) protocol was established. For this method, instead of purifying whole translatomes via ultracentrifugation of elongation-inhibited cell lysates, ribosomes bound to a specific translation co-factor are enriched via immunoprecipitation and simultaneous nuclease digest of the lysate. Subsequent isolation of RPFs from the purified ribosomes allows to identify the transcripts whose translation depend on physical interaction of the ribosome with the factor of interest (Becker et al., 2013; Oh et al., 2011). cpSRP54 is known to be involved in post-translational import of light harvesting complex subunits into the chloroplast (Ziehe et al., 2018). However, apart from this function it is also known to bind to chloroplast ribosomes in higher plants and is suspected to direct these to the thylakoid membrane if a respective target protein is translated, similar to its cytosolic or bacterial counterparts which are directing ribosomes to the endoplasmatic reticulum or the bacterial plasma membrane (Hristou et al., 2019; Nagai et al., 2003).

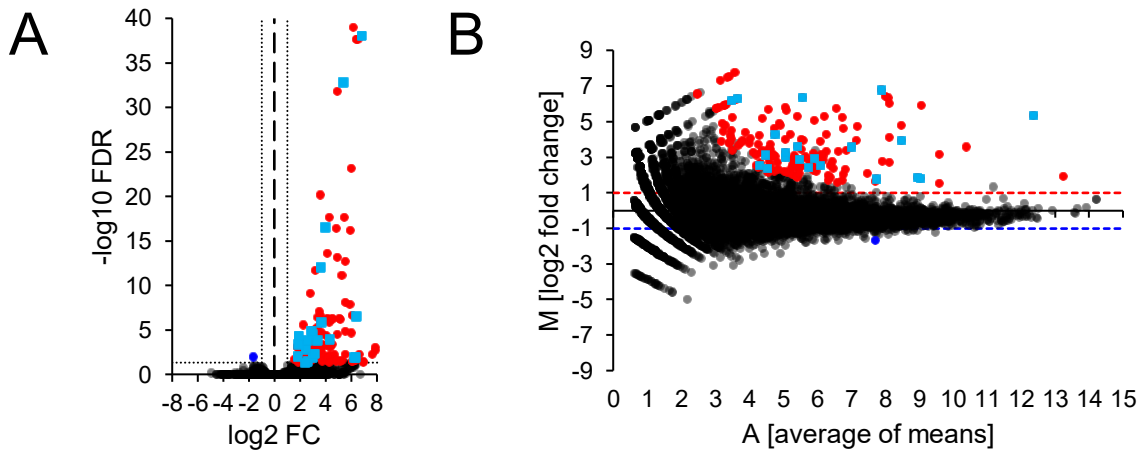


Figure 3-25: DE-analysis of SRP54-selective ribosome profiling experiment. (A) Volcano plot representing transcripts in seRP experiment by Benjamini-Hochberg corrected p-values (-log10 FDR) versus log2 fold change (log2 FC) of ctrl / induced samples. Black dots represent transcripts not differentially expressed on the translational level, blue dots represent depleted transcripts, red dots represent enriched transcripts without annotated chloroplast localization. Cyan dots represent transcripts enriched with annotated chloroplast localization. Dashed vertical line indicates 0 change, dotted grey lines represent borders of significance on both axes. (B) MA-plot representing transcripts in seRP experiment by their log2 average expression between both conditions and their log2 fold change. Colors like A. Blue and red dotted lines represent the borders of significance on the log2-fold change axis. All values were normalized by the LOWESS procedure.

To establish the seRP technique, a pilot experiment was run targeting the chloroplast SRP54 homolog in biological duplicates. *C. reinhardtii* cultures were grown under photoautotrophic conditions in day/night cycles in photobioreactors (see Table 2-2). After several cycles, when the cultures reached the desired density, cells were harvested 2 h after the onset of illumination. These conditions were chosen as attempt to increase translational activity in the chloroplast and hence increase the chance to sample ribosome-bound cpSRP54. cpSRP54-ribosome complexes were purified in a pulldown with a cpSRP54-specific antibody immobilized to agarose beads. The pulldown data then was compared against conventional translatomes sampled according so protocol iii from the same cultures.

In an initial quality check (see Supplementary Figure 12 and Supplementary Figure 13), the samples were found to exhibit a clear peak at 31 nt length of CDS-derived cytosolic RPFs while the distribution of chloroplast RPFs was much broader. In the pulldown samples the clear shape of the cytosolic RPF length distribution was preserved while the distribution of chloroplast RPFs changed considerably, covering the whole range of 20 to 39 nt. Surprisingly, the pulldown samples also showed a clear 27 nt peak of 5'-UTR derived RPFs in the chloroplast. All samples were found to have a clear periodic pattern regarding their frame preference of cytosolic RPFs with an in-frame preference of ~60%, confirming the ribosomal origin of purified fragments. However, determination of in-frame preference of chloroplast RPFs failed due to a low number of RPF's mapping to start codons which lead P-site offset calculation to fail.

To identify putative targets of cpSRP54, RPFs purified in the seRP experiment were compared to total translatomes prepared from the same cultures in a differential expression analysis using the DESeq2 R-package (Love et al., 2014).

Especially among the weakly expressed transcripts, the log2 FC values were slightly skewed towards the enriched samples, as seen by a LOWESS curve (Cleveland, 1981) calculated on the data set (Supplementary Figure 10). To correct for this technical bias, the log2 FC values of the data set were normalized by the LOWESS curve (Figure 3-25 B). As expected for an enrichment experiment, many more transcripts were identified as significantly enriched than as depleted (149 enriched versus 1 depleted) (Figure 3-25 A and B). The only transcript found to be depleted was *Cre02.g107150* (log2-fold change: -1.66, adjusted p-value: 0.01), encoding an uncharacterized protein predicted to localize to mitochondria. With log2 FC values of up to 7.74 (resembling ~214-fold enrichment) and a median log2 FC of 3.39 (~10-fold enrichment), enrichments were generally very strong.

Table 3-2: Transcripts enriched in cpSRP54-selective ribosome profiling encoding proteins known or predicted to be chloroplast localized. Transcripts are sorted according to enrichment strength in decreasing order. FDR represents the Benjamini-Hochberg corrected p-values.

#	transcript name	log2-fold enrichment	FDR		#	transcript name	log2-fold enrichment	FDR
1	**ARS6**	6.79	7.54E-39		11	**Cre12.g497800**	2.98	1.13E-02
2	**Cre06.g278153**	6.39	2.74E-07		12	**Cre04.g226200**	2.94	2.81E-05
3	**Cre16.g676750**	6.33	1.00E-02		13	**Cre10.g451600**	2.85	1.18E-05
4	**Cre03.g156650**	6.22	1.15E-02		14	**Cre02.g090950**	2.58	3.35E-02
5	**SRP54L**	5.36	1.30E-33		15	**Cre10.g464950**	2.53	1.22E-04
6	**Cre08.g800961**	4.32	1.02E-04		16	**Cre07.g323700**	2.43	1.53E-03
7	**Cre06.g800753**	3.63	1.27E-06		17	**orf854**	2.40	4.77E-02
8	**Cre11.g467562**	3.60	9.01E-13		18	**EFG10**	1.88	4.37E-05
9	**Cre14.g621501**	3.28	1.22E-04		19	**LCI15**	1.83	3.88E-04
10	**Cre12.g801441**	3.13	4.43E-03		20	**SULTR3**	1.78	8.94E-03

However, *orf528* was only chloroplast encoded transcript identified among the significantly enriched candidates. Overall, among the enriched transcripts, 20 were known or predicted to encode chloroplast localized proteins (Table 3-2, represented as cyan dots in Figure 3-25 A and B), by the protein localization prediction tool PredAlgo (Tardif et al., 2012), including the transcript of cpSRP54 itself (5.36 log2 FC, resembling ~41-fold enrichment). The remaining enriched transcripts (red dots) instead were known or predicted either to encode non-chloroplast proteins or proteins with unknown localization due to insufficient predictability (67 of 129 or 52%, see Supplementary Table 1). This large number might suggest that a much greater proportion of enriched transcripts could encode chloroplast proteins, especially since most of them, as well as most of the candidates classified as chloroplast-localized, are not characterized to date. Of note, some of the enriched transcripts are also known to encode non-chloroplast proteins (e.g. cytosolic MARS1 or mitochondrial MRPS26) and therefore clearly have to be considered contaminants. The outcome of the experiment is surprising since cpSRP54 is chloroplast-localized and therefore should not be able to bind cytosolic ribosomes and enrich their footprints in an IP-experiment. According to the commonly accepted model, chloroplast protein import occurs post-translationally through the chloroplast TIC-TOC pore complex (Nellaepalli et al., 2023). However, an additional co-translational import mechanism of nuclear-encoded chloroplast proteins was recently proposed and could explain the enrichment of the footprints of such transcripts (Sun et al., 2024). In such a scenario, the polypeptide chain could be channeled through the pore complex and cpSRP54 could receive the N-terminus of the co-translationally imported protein at the stromal side of the chloroplast inner envelope in a similar fashion as it is known to do in case of the post-translational import of LHC-proteins (Ziehe et al., 2018). Interestingly, among the enriched candidates annotated to encode chloroplast-encoded proteins, *Cre02.g090950* is annotated to contain an ankyrin-repeat domain. In higher plants, cpSRP54 was reported to bind cpSRP43, its partner for LHCP import into chloroplasts, via interaction with its ankyrin-repeat domain (Ziehe et al., 2018). Not a single LHCP-encoding transcript was found among the enriched candidates, suggesting that

even if cpSRP54 may be involved in a co-translational import mechanism, LHC-subunits may be imported exclusively in post-translational state.

3.3.2 A local enrichment within its ribosome profile suggests *psaB* to be a substrate of cpSRP54's co-translational action in *C. reinhardtii*

The absence of chloroplast encoded transcripts among globally enriched candidates indicates that a putative co-translational interaction of cpSRP54 with the chloroplast ribosome in *C. reinhardtii* might follow a different mode than initially assumed. Considering that cpSRP54 might bind only transiently to the chloroplast ribosome during translation of specific protein regions or rather their emergence from the polypeptide exit tunnel, the experiment would yield partially enriched regions within the ribosome profiles of cpSRP54's target transcripts but not necessarily a global enrichment of these candidates. To further investigate this idea, the data set was re-analyzed by normalizing and averaging ribosome profiles built from RPFs enriched in cpSRP54-immunoprecipitations and comparing them to corresponding reference profiles built from the total translatome. By manually inspecting all profiles of chloroplast encoded transcripts, the *psaB* profile was found to display a striking, albeit narrow peak indicating a putative enrichment of RPFs encoding approximately codons 105 to 120 (Figure 3-26 A, Figure 3-27 A). Considering the polypeptide exit tunnel of the ribosome shielding 30 to 40 amino acids (Chadani et al., 2021), this region is decoded when *psaB*'s first transmembrane domain (encoded by position 47 to 70) emerges from the ribosome (Figure 3-27 B). Apart from *psaB*, *ftsH* was the only other chloroplast encoded transcript that showed a prominent enrichment in a very short region encoding codons 1464 to 1468 (Figure 3-26 D, red box). Other candidates, like *psbA*, which in higher plants represents a main target of cpSRP54's co-translational action (Hristou et al., 2019), only showed low local enrichments (Figure 3-26 C). However, compared to *psaB*'s case, their enrichments are weak, further experiments are needed to confirm these as targets. These findings might indicate that the strength of cpSRP54's association to nascent chains may differ across its co-translational targets. Furthermore, its role for membrane insertion of chloroplast encoded proteins in *C. reinhardtii* may be specifically important to *psaB* and *ftsH.* Interestingly, *psaB* is also the top-level subunit in photosystem I's CES-cascade that orchestrates the stoichiometric synthesis and membrane insertion of PS I subunits (Wostrikoff et al., 2004). The enrichment profile of *psaA* however (Figure 3-26 B), which is the subunit following *psaB* in the cascade, shows no signs of cpSRP54 binding, indicating that cpSRP54's role in PS I biogenesis may mainly apply to *psaB*.

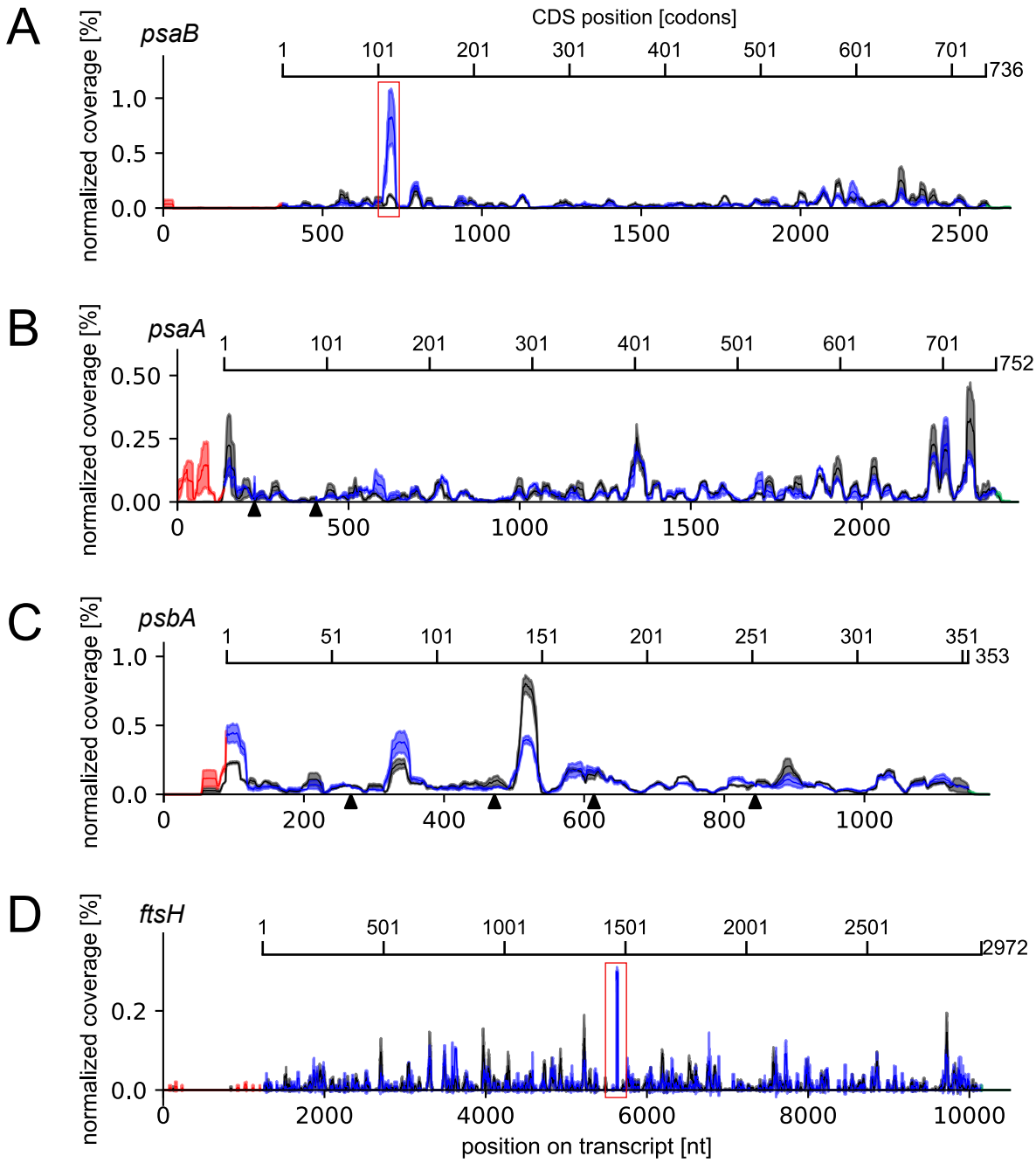


Figure 3-26: Comparative ribosome profiles of chloroplast transcripts. (A – D) Transcript names are indicated in italic font above the y-axis. Black lines represent normalized and averaged reference profiles from translatome samples. Colored lines represent the normalized and averaged profiles of RPFs enriched by cpSRP54-immunoprecipitation. Red parts represent 5'-UTRs, blue parts represent coding sequences and green parts represent 3'-UTRs. Shaded areas represent the standard deviation-based error bands. Scale bars on top indicate the codon-wise position along the coding sequence (CDS). Black arrows indicate exon-exon borders. Red box marks the enrichments within *ftsH*'s and *psaBs* profiles.

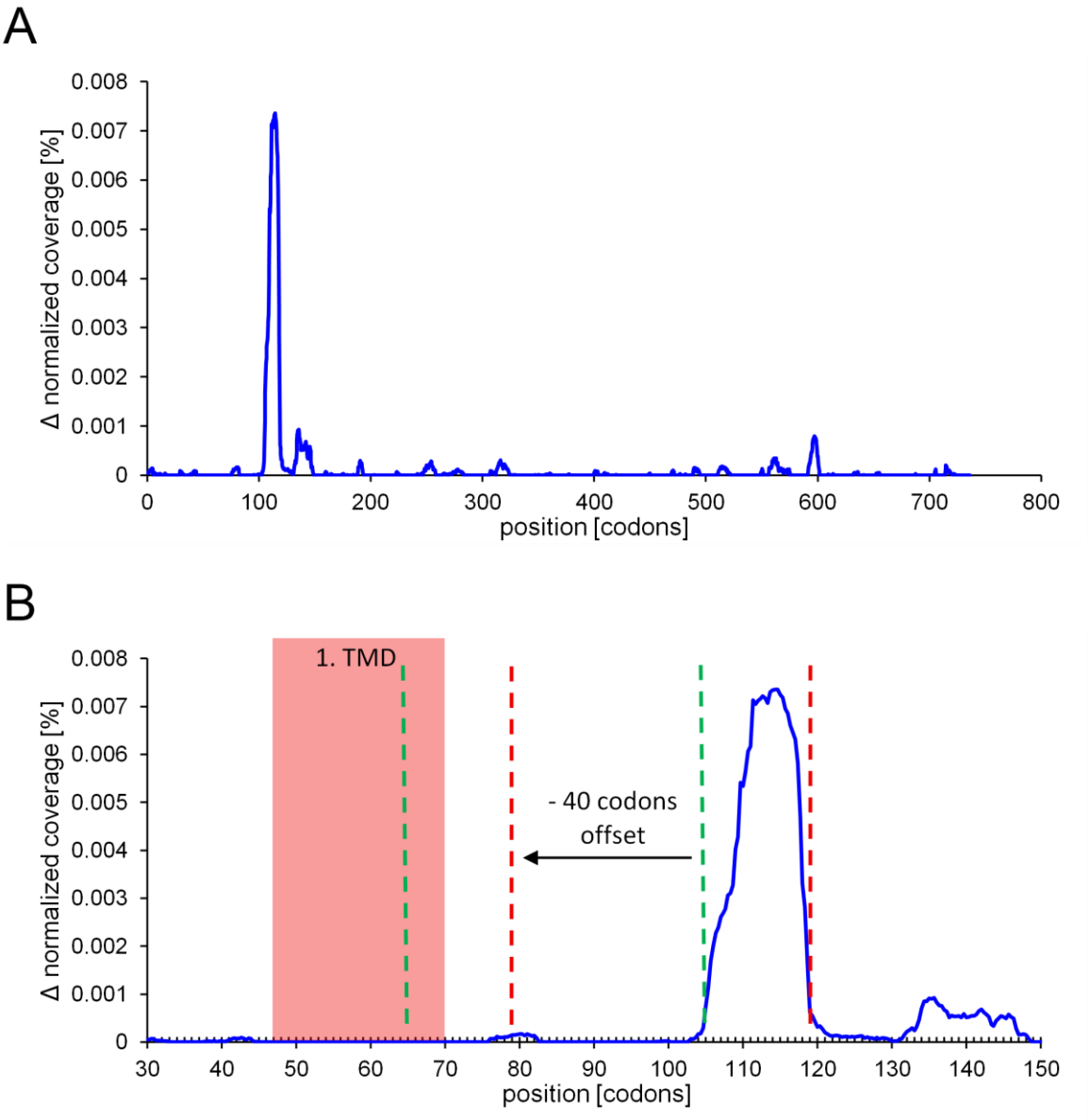


Figure 3-27: Difference plots of normalized coverage (enrichment – control). (A): Difference profile along the whole *psaB* transcript. (B): Details of enriched *psaB* region and positional comparison to the location of PsaB's first transmembrane domain (TMD). Arrow indicating the presumable offset between the enriched region and the region encoding the part of the polypeptide emerging from the ribosome when a putative binding of cpSRP54 takes place. Green and red dashed lines enclose the region of RPF-enrichment on the transcript and the region encoding the part of the nascent chain presumably emerging from the ribosome at the time when the enriched region is translated. Red shaded area represents the region of the transcript encoding the first transmembrane of PsaB. To calculate the enrichment profiles, the average normalized coverage of the control samples was subtracted from the average normalized profile of the pull-down sample. Positions with values smaller than zero were set to zero.

4 Discussion

4.1 RSf provides Ribo-Seq tailored workflows while maintaining maximum flexibility in analysis

Since publication of the Ribo-Seq method in 2009 (Ingolia et al., 2009), numerous bioinformatic tools have been developed that address the very specific challenges of quantitatively analyzing Ribo-Seq data or provide quality control workflows for the data (reviewed up to 2020 in Kiniry, Michel and Baranov, 2020). Many of these tools are accessible online through the Galaxy online platform (Afgan et al., 2022; Legendre et al., 2015), enabling also non-bioinformaticians to perform complex data analysis tasks. However, most established tools aim at providing “full workflow” solutions that prompt the user for data input and execute a script that outputs the result of a static workflow. While this simplification is generally desired to enable non-specialists to analyze bulk data, it is also a limitation to the advanced user’s capability to interact with the data and adapt an analysis to the specific question of an experiment. Another problem in the specific case of analyzing organellar translatomes is the circumstance that most established tools do not distinguish between RPFs of different organellar origin. RSf was developed as a toolkit to overcome these problems in order to analyze the three translatomes of *C. reinhardtii* separately and provide the user full flexibility in analysis, preferably by using the functions in exploratory data analysis. Similar to Plastid (Dunn & Weissman, 2016), a Python package designed for more general NGS-analysis, RSf is built on the Scientific Python stack (Virtanen, Gommers, Oliphant, Haberland, Vázquez-Baeza, et al., 2020), ensuring that all analysis results can be seamlessly transferred to custom downstream analyses build on packages of the common Python data science ecosystem. Different than most established tools, RSf combines both a variety of quality control functions and deep analysis functions to determine expression strength. For example, RSf provides options to calculate organelle-wise RPF length distributions and tri-nucleotide periodicity similar to the QC tool Ribo-seQC (Hsu et al., 2016), but also enables the user to calculate RPKM, CPM and read counts transcriptome-wide either considering coding regions only (for Ribo-Seq data) or full transcripts (to allow also RNA-Seq quantification) similar to using the count function of the STAR aligner (Dobin & Gingeras, 2015). Instead of performing a static workflow, the user is provided the ability to combine RSf’s functions in every desired way. A special trait of RSf is its object-oriented design that treats every transcript to be analyzed as a separate object that stores all important information connected to the transcript (e.g. number of mapped reads, distribution of reads across the transcript, etc.). In this way, this information can always be accessed upon request after calculation, which allows to perform highly flexible and deep analyses to perform

calculations on specified sets of transcripts, circumventing the necessity to perform a genome-wide analysis if only specific transcripts are of interest. An exclusive feature of RSf not provided by comparable tools is the extraction of ribosome profiles for transcripts of interest, which enable the user to investigate translational dynamics on the level of specific transcripts, e.g. by comparing these profiles against specific sequence contexts or complementary data from other experiments. All Ribo-Seq coverage data generated in this work was provided to the scientific community by making it accessible online via the JGI Phytozome platform (Goodstein et al., 2012) to enable scientists to utilize this valuable information for their own analyses. However, despite RSf's ability to provide deep insights into Ribo-Seq data sets, it is not able to substitute for every established tool. For example, while RSf can deliver genome-wide expression values calculated on CDS basis, it does not contain a function to perform a statistically sound differential expression analysis. Instead, depending on the experiment and the questions to be answered, the analyst needs to assess if RSf alone is suitable or if results obtained from RSf need to be passed on to another tool afterwards.

4.2 Quantification of the translatome reflects the lifestyle of *C. reinhardtii* cells

In this study, the translatome of mixotrophically cultivated *C. reinhardtii* was quantified and analyzed in great depth. As presented in Figure 3-5, special attention was paid to the examination of the 150 top-translated transcripts. The translatome sampled under these conditions is dominated by candidates involved in anabolism, specifically with transcripts related to photosynthesis and such encoding ribosomal proteins making up for most of the 150 top-translated transcripts. Considering the typically high concentrations of these candidates' protein products in the cell, it is not surprising to find them among the top-translated transcripts. For example, LHCII alone is estimated to account for roughly one third of the total thylakoid protein of a higher plant's cells (Chitnis & Thornber, 1988). Similarly, ribosomes are estimated to account for 30% of a cell's total dry weight in *E. coli* (Soler-Bistué et al., 2015) and presumably reach a high percentage in *C. reinhardtii* as well.

Besides these examples of proteins that account for large parts of the cell's biomass, other proteins were identified to be highly expressed that are usually not primarily known for a high impact on the biomass of a cell but are clearly functionally related to anabolism by their involvement in the glyoxylate cycle. The glyoxylate cycle is a metabolic pathway that converts C2 acids to C4 acids for gluconeogenesis and is widely distributed among organisms (Kornberg & Krebs, 1957; Schnarrenberger & Martin, 2002). In *C. reinhardtii* it was found

essential for the metabolization of acetate under hetero- or mixotrophic growth conditions (Plancke et al., 2014). The presence of key enzymes of the glyoxylate cycle (ICL1 – isocitrate lyase, PCK1 – phosphoenolpyruvate carboxykinase, MDH1 – malate dehydrogenase), acetate transporters (GFY3-5) and the acyl-carrier protein ACP2, involved in fatty acid synthesis, among the top-translated transcripts appears a consequence of the clearly anabolism-favoring growth conditions of the cells and reflects the respective metabolic response (Durante et al., 2019; Msanne et al., 2021; Plancke et al., 2014).

As highlighted in Figure 3-6 A, only a fraction of the annotated transcripts in *C. reinhardtii* account for a great share of RPFs with the top 150 translated transcripts being accountable for approximately 30% of RPFs. The translational efficiency of these transcripts showed that a high translational activity does not necessarily coincide with a high transcript accumulation. Instead, there is no clear pattern observable. However, it should be noted that this conclusion is based on a few assumptions about the data. First, it must be mentioned that Ribo-Seq data is of fundamentally different nature than RNA-Seq data since it is derived from sampling a highly dynamic process and therefore represents only a snapshot of translation. The way this data is interpreted in this study assumes that this snapshot reflects a steady state of a transcripts' continuous translational activity which is in equilibrium with the transcripts' concentration and the half-life of the encoded protein. However, it is questionable if such a model applies to all transcripts. It is very likely that translational regulation is more complex and may integrate additional aspects. Translation can be impacted by various factors like the availability of tRNAs, translation (co-)factors or formation of secondary structures within translated mRNAs and their resolution which all may change the mode of a transcript's translation only in a matter of seconds. Due to the lack of knowledge of all these factors and their influence, the very dynamic nature of translation is mostly neglected by the way the data is interpreted in this study and others. Instead, the data is handled very similarly to common RNA-Seq data, which is much more static in its nature since it only reflects the abundance of a transcript at a given timepoint. The concentration of a transcript within a cell is also subject to change. However, the change may not appear as drastic as in translation, where the translation initiation of a transcript potentially could be stopped completely from one moment to the other, for example due to the action of transcript-binding co-factors that may suppress translation (Babitzke et al., 2009). The transcript titer at a given time point is always dependent on the abundance at a previous time point in that it increases or decreases gradually due to synthesis or degradation until a new steady state is reached. This relationship does not necessarily apply to the translational status of a transcript. Another problematic aspect in the interpretation of Ribo-Seq data is the uncertainty whether a ribosomes' footprint is sampled from an actively translating ribosome or from a stalled or paused ribosome. Potentially, ribosomes could stop translating a transcript at a certain time point without releasing the

transcript molecule or vice versa, they may bind to the initiation region of a transcript and dwell at this position without entering elongation. Such events may inflate the RPKM value measured for these transcripts. Time resolved studies with initiation inhibitors like harringtonine (or lincomycin for the chloroplast) and subsequent measurement of ribosome run-off over short time intervals could circumvent this problem to assess the true translation rate for each transcript (Chotewutmontri & Barkan, 2018; Ingolia et al., 2011). Since every ribosome is expected to produce one protein molecule during translation of a transcript, the number of molecules synthesized within a given time span can be assessed by inhibiting initiation for this time span and quantifying the loss of RPFs per time for every transcript. Such an approach would include the effects of pausing, codon specific dwell times and individual regulation etc. on the translation rate of every transcript. Moreover, to investigate the topic of a translational steady state, it would be necessary to carry out such experiments in narrow time intervals on cultures with synchronized cell cycles by growing them in day / night cycles. With such a strategy, all cells of a culture should be always in the same state, sharpening the accuracy of the sampled data and instead of providing “averaged” measuring results, it would allow to observe short time changes as well.

A detailed functional enrichment analysis was carried out manually on the 150 top-translated transcripts due to a general lack of GO-term annotations (Gene Ontology Consortium, 2004) for many genes in the *C. reinhardtii* genome. Although the result was a very informative summary of the functional roles of the transcripts investigated, this procedure is not suitable for a whole-translatome analysis. Modern high-throughput methods are able to generate vast amounts of data in short time, which necessitates the option for quick exploratory data analysis. The GO-enrichment analysis is a very important tool to draw quick conclusions from the data. This highlights that the lack of GO-terms annotated in the *C. reinhardtii* genome poses a major problem for modern high throughput methods and that improvements on the annotation would be highly beneficial for the field. Even though experimental data may be limited for *C. reinhardtii*, such a task could be aided by the comparison of protein sequences and protein names with other species and inferring GO-terms based on such comparisons.

Despite most of the 150 top-translated transcripts encoded well-known proteins, they also included 6 uncharacterized transcripts (LCI3, Cre01.g051500, Cre13.g579450, POC8, Cre10.g452350 and COP21) that may represent interesting candidates for closer investigation.

4.3 Translational regulation in the cytosol of *C. reinhardtii* may be more common than previously thought

As described above, the mixotrophic translatome of *C. reinhardtii* mainly reflects the cultures' lifestyle with transcripts encoding proteins involved in anabolism dominating the top 150 expressed candidates. However, apart from these unsurprising candidates, others represent very interesting objects for further investigation. The VIG1 (vasa intronic gene 1) transcript (124th highest translated) is one of those examples. VIG1 protein was found to bind to both, polysomes and the RNAi-mediating RISC (RNA-induced silencing complex) in *C. reinhardtii* via physical interaction with the RISC component AGO3. To date, this mechanism was only observed in *C. reinhardtii*, suggesting that the influence of RNAi on cytosolic translation in the green alga may be especially important. Furthermore, the interaction mediated by VIG1 was observed not to cause a destabilization of translated mRNAs (Ma et al., 2020), suggesting it represents a mechanism for translational regulation. As suggested by literature and by the quantitative Ribo-Seq data in this work, VIG1 is a strongly expressed protein with titers similar to *bona-fide* ribosomal proteins found in *C. reinhardtii*. Therefore, it is expectable that translational regulation through VIG1 might be a very common event in the cytosol. A study on small RNAs predicted 6,164 potential small RNA loci in *C. reinhardtii* based on RNA-Seq data (Müller et al., 2020), exposing an enormous arsenal of potential regulators of gene expression that could possibly act via RISC-VIG1 interaction at the stage of translation, even when expecting only part of the predicted loci to encode true sRNAs. Although mainly known for the targeted degradation of transcripts, it is widely accepted that some RNAi mechanisms specifically act on the level of translational regulation without affecting mRNA stability (Eulalio et al., 2008; Iwasaki et al., 2009; Pratt & MacRae, 2009). Moreover, advances in the last years indicate that RNAi in *C. reinhardtii* is much more complex than previously anticipated and differs from the typical scheme that is assumed for higher plants. Ribosome profiling in a knock-out mutant of DCL3, a factor essential for miRNA processing, identified 35 transcripts whose translation is regulated by miRNA-mediated RNAi (B. Y. W. Chung et al., 2017). This low number is in great contrast to the number of sRNA loci proposed (Müller et al., 2020), potentially suggesting that a much greater portion of transcripts might be translationally regulated via the far less studied siRNA-mediated RNAi pathway. Generally, siRNA-mediated RNAi was also shown to act on translation at the post-initiation stage in *C. reinhardtii* (Ma et al., 2013). A detailed study of the proposed sRNA loci in combination with Ribo-Seq could potentially shed light on this complex network of post-transcriptional regulation.

In this study, 3,346 transcripts were identified to have extreme initiation peaks, indicating widespread pausing of ribosomes on the start codon between the initiation and elongation stage. These transcripts would represent interesting candidates to study in the context of siRNA-mediated translation regulation. The relative height of these peaks was shown to be independent from the predicted transcripts' propensity to form secondary structures within the initiation region, suggesting that sterical hindrance of ribosome translocation by such structures may not be the cause of such extreme initiation peaks. However, not only RNAi, but other mechanisms as well should be considered to cause this phenomenon. In mammalian cells, it was shown that m^6A mRNA methylation (methylation of adenosine residues) near the start codon is a regulator of translation by inhibiting elongation start after the ribosome has been assembled on a start codon, yielding large accumulations of RPFs within the initiation region (Dong et al., 2021). The authors of the study showed that this mechanism is an important suppressor of tumorigenesis in mammals due to the limitation of translation output and therefore the prevention of uncontrolled growth. m^6A RNA methylation was also observed in *C. reinhardtii* (Lv et al., 2023) and other plant species (Li et al., 2014; Luo et al., 2014). Although the main methylation sites in these species were mapped to stop condons and 3'-UTRs, in all cases a smaller accumulation of m^6A sites close to annotated start codons was found as well. These findings suggest that m^6A-mediated ribosome pausing at start codons may be a conserved mechanism between kingdoms and a very promising candidate to explain the occurrence of extreme initiation-peaks in *C. reinhardtii* and potentially other species of the green lineage. While gene expression in the chloroplast has been proposed to be mainly post-transcriptionally (and widely translationally) regulated since a long time now, less attention has been spent on mechanisms of translational regulation in the cytosol. In this work, clear indicators were found that the extent of translational regulation of gene expression may have been underestimated in the cytosol of *C. reinhardtii* and that the topic might represent an interesting future research object.

4.4 Chloroplast translation appears highly specialized in *C. reinhardtii*

As exemplified in Figure 3-8, the gene expression profile of the chloroplast is drastically different from the cytosol. While both, transcript abundance and translation generally appear elevated compared to the cytosol, another striking difference is that the ratio of both is shifted towards the transcript abundance. Due to this fact, it is very likely that translation represents the limiting component for effective gene expression in the chloroplast and therefore the

controlling factor. The translational control becomes evident when studying the distribution of RPKM values of functionally or mechanistically related transcripts on the translational and transcript level. As seen in Figure 3-10, such groups of transcripts can differ widely on the transcript level and still have a very narrow variation on the translation axis. In the case of the chloroplast ATPase, the translational strength of the complex subunits even roughly reflects the stoichiometry of the subunits in the assembled protein complex (Chaux et al., 2023). In case of PS II subunits, an astonishing range of transcript abundances was observed that may potentially reflect the existence of a “buffer contingent” for specific transcripts. Former studies showed that the half-life of chloroplast transcript *psbA* ranges between 5 and 10 hours in spinach, which is exceptionally long as compared to the average transcript half-life in bacteria and yeast (Klaff & Gruissem, 1991). In *E. coli*, the median transcript half-life was determined to range between 2.8 and 4.2 min, depending on the growth conditions and a median transcript half-life of 32 min was measured in yeast (Esquerré et al., 2014; Geisberg et al., 2014). Apart from the example of *psbA*, chloroplast transcripts in general were found to be exceptionally long-living, supporting the idea of a large translationally inactive transcript pool (Zoschke & Bock, 2018).

The general RPF length distribution of the chloroplast differed greatly from the cytosol in exhibiting three major RPF length species instead of one. A closer look at the transcript specific RPF length distributions for all chloroplast transcripts revealed that many transcripts yield very distinct RPF length distributions. It was speculated that these distributions might be heavily influenced by pausing ribosomes that may yield shorter RPFs possibly through being arrested in a rotated post-translocation state during elongation as exemplified by the ribosome profiles of *atpH* and *psbA*. The general chloroplast RPF length distribution suggests that such pausing sites are widespread in the chloroplast and may represent evolutionary adaptations to the needs of translating a very small distinct selection of transcripts. Ribosome pausing in the chloroplast has been reported before, for example on protein level during translation of *psbA* in barley and on the whole-plastome level by ribosome profiling in *A. thaliana* (Gawroński et al., 2018; Kim et al., 1991). The predominant putative pausing site of *psbA* found in this work was not observed in the earlier study in *A. thaliana*. However, since the authors of the study filtered out all RPFs shorter than 28 nt, they may unintentionally have removed especially paused ribosomes’ RPFs from their data set. Interestingly, the predominant putative pausing site within *psbA* was found earlier to be important in the context of OHP2-dependent chlorophyll binding to D1 protein, which may be a process that could benefit from translational pausing (F. Wang et al., 2023). In higher plants, *psbA* was found to be incorporated co-translationally into the thylakoid membrane into a complex with D2 protein, giving rise to a 17 kD D1-intermediate that is complexed by D2 (L. Zhang et al., 1999). The authors of the study estimated that this intermediate product might occur due to ribosome pausing roughly when

the third transmembrane domain emerges from the ribosome, which is close to the putative pausing site found in this study (emergence of the second TMD from the ribosome). Altogether, it can be assumed that *psbA* translation is paused when the second transmembrane helix emerges from the ribosome, resulting in an accumulation of short RPFs covering codons 141 to 150. Then the nascent chain may be inserted co-translationally into the thylakoid membrane upon OHP2-dependent binding of chlorophyll to the molecule.

Apart from widespread pausing events, the transcript specific RPF length distributions observed in the chloroplast may suggest the existence of dedicated ribosome pools that specifically translate a subset of chloroplast transcripts. Due to the low number of genes encoded on the chloroplast genome and the partly vast extents in which specific proteins are synthesized, the situation in the chloroplast and therefore its requirements regarding its protein synthesis machinery are fundamentally different from the cytosolic situation. In the cytosol, an enormous number of different transcripts has to be translated at mostly moderate rate, possibly necessitating a more generalized protein synthesis machinery that yields rather fixed RPF lengths due to less individual fine-tuning. This difference between the two systems might evolutionarily favor a specialization of the ribosome in the chloroplast to optimize the production of specific proteins. A good example for a protein synthesized possibly by a specialized ribosome pool may be AtpH. This subunit of the chloroplast ATPase has several characteristics that might generally pose a challenge for its synthesis: It is produced in large quantities, is part of a membrane protein complex and is very short (43 amino acids). AtpH only represents one candidate of many proteins encoded on the chloroplast genome that have similar characteristics and very likely come with very specific requirements and challenges for proper translation which could create significant evolutionary pressure towards ribosome specialization in the chloroplast.

4.5 Chloroplast translational capacity may be controlled by the titer of two specific ribosomal proteins

An interesting aspect of the translatome analyses in this work was the observation that in total 58 transcripts encoding cytosolic ribosomal proteins were among the 150 top-translated proteins, but only one single chloroplast ribosomal protein (PRPL7, 52nd strongest translated). The bacterial homolog of this protein, bL12, is known to be essential for proper ribosome function as it represents a binding site for multiple translation co-factors (Miyoshi et al., 2009). It was hypothesized that ribosomes in most bacteria and chloroplasts can bind up to six copies of the protein as dimers and that its abundance directly influences elongation speed in *E. coli*

by altering elongation factor binding dynamics at the ribosome (Davydov et al., 2013; Hofmann et al., 2025). With its important role for ribosome function, PRPL7, or corresponding to the new nomenclature, bL12c (Ban et al., 2014), represents a very promising candidate for regulating the chloroplast translation by controlling the titer of bL12c protein synthesized in the cytosol and transported to the chloroplast. Under vivid growth conditions given in the context of this work, it would make sense to eliminate a potential bottleneck for chloroplast translation by over-accumulating bL12c protein enabling unhindered chloroplast biogenesis and division. This together with the need to enable dimerization of the protein could explain why PRPL7 is expressed at much higher rate on the translational level than other chloroplast ribosomal proteins. Another interesting finding is the fact that while chloroplast ribosomal proteins encoded on the chloroplast genome generally are translated at very similarly high levels, Rpl20 is very poorly translated. In bacteria, this protein is an important factor in ribosome biogenesis where it is a first-stage binder of the 23S rRNA involved in proper rRNA folding. The protein was found to repress its own translation by binding *rpl20* transcript, and its absence was found to limit the translational capacity in bacteria dramatically (Haentjens-Sitri et al., 2008; Raibaud et al., 2003). It is conceivable that the importance of Rpl20 for the rRNA folding process is conserved in the chloroplast and is utilized to control ribosome biogenesis. Together, PRPL7 and Rpl20 could both represent two potent regulators for chloroplast translation on different levels. One on the level of ribosome biogenesis regulating the titer of ribosomes in general (Rpl20) and the other one on the functional level by controlling the titer of ribosomes available for translation and regulating the binding kinetics of elongation factors (PRPL7).

4.6 uS11c is essential for translation of four specific chloroplast proteins

Translation was investigated in an inducible knock-down mutant of the chloroplast S11 homolog, a subunit of the chloroplast ribosomal SSU, in induced versus non-induced state. Results of previous experiments revealed that the phenotype of the knock-down is associated with intensive morphological changes affecting the appearance of the chloroplast and extensive formation of palmelloid colonies. Analysis of the Ribo-Seq data revealed that these morphological changes are accompanied by an increase of translation of cell wall proteins and enzymes that may have a role in the remodeling of the cell wall, contributing to palmelloid formation. These enzymes may represent interesting targets for further studies. It was previously shown that uS11c depletion results in strong reduction of photosystem II subunits D1 and D2. However, for neither of both a reduced translation was observed, indicating that

the proteins are translated normally, but are degraded afterwards. In contrast to these subunits, translation of *psbC* was clearly reduced by -1.35 in Log2-space (resembling ~40% residual translation). In other studies, it was shown that reduced *psbC* translation leads to a strong reduction of proteins D1 and D2 as well (Zerges et al., 1997), suggesting that the same mechanism is the underlying cause for their loss upon uS11c depletion since they are not affected on the translational level directly. PS II biosynthesis is regulated via a CES cascade in which the synthesis of PS II core subunits of lower hierarchy is directly coupled to the availability of the higher-level subunits (Minai et al., 2006). However, PsbC was found not to be a CES subunit, but it was hypothesized that a lack of the subunit results in proteolytic digest of the PS II assembly intermediates, once more explaining the loss of D1 and D2 in the mutant (de Vitry et al., 1989; Rochaix et al., 1989). Interestingly, no PS I subunits were observed to be translationally downregulated by uS11c depletion. This would result in an imbalance between PS II and PS I that might result in impaired CO_2 fixation. A recent study suggested that palmelloid formation might be a natural response to high light stress in *C. reinhardtii* that is supposed to reduce light intensity due to the cells shading each other (Suwannachuen et al., 2023). The same mechanism might cause the palmelloid phenotype upon uS11c-depletion since the inability to produce PS II subunits may result in light-induced stress at much lower intensities than usual. Hence, shading could also be an efficient mitigation strategy. Of note, uS11c depletion was found to cause a growth deficit under standard (50 µE) and especially under high light illumination (800 µE) in a previous study (Gotsmann, 2018). While the formation of palmelloid colonies would represent a secondary effect in response to uS11c depletion, the primary effect is much more difficult to recognize. In general, the depletion of a ribosomal protein would be expected to have a negative impact on translation of all transcripts. However, apart from *rps11* itself, the depletion of uS11c only reduced translation of *psbC*, *atpH*, *psbK* and *ycf12*. Notably, none of the affected candidates showed a significant change in transcript abundance, suggesting their change on the translational level is not a consequence of changes on the transcript level and therefore the candidates truly represent uS11c-dependent transcripts. Interestingly, all these transcripts contain SD(-like) sequences in their 5'-UTRs. Inspection of the ribosome profiles of these transcripts revealed that *atpH* showed strong changes in its translation dynamics under uS11c depletion, especially around the start codon with a higher coverage directly upstream of the start codon and a reduced coverage at the start codon itself. Since uS11c is located very close to the anti-SD cleft of the SSU, and *atpH* has a perfect SD-sequence in its 5'-UTR, it is likely that the depletion of uS11c causes a problem with initiation of *atpH*. Another striking difference was found around codon 41. This site also shows an accumulation of RPFs in the reference profile, suggesting it might represent a pausing site. The polypeptide exit channel of the ribosome was found to fit approximately 30 to 40 amino acids in it before the nascent chain emerges from the ribosome (Chadani et al.,

2021). Since AtpH is an integral membrane protein, it might be inserted into the thylakoid membrane directly upon translation, possibly requiring the ribosome to pause for this task. It is possible that this is promoted by a factor whose release from the ribosome requires interaction with uS11c to continue elongation after the pausing. Interestingly, *psbK* and *ycf12* transcripts did not exhibit changes in their translation dynamics, indicating that their downregulation in response to uS11c depletion is caused only by a diminished initiation propensity and their residual expression is solely attributable to the remaining uS11c molecules after 48 h of induced knock-down. Since they also harbor SD-like sequences in their 5'-UTRs, but not perfect SD-sequences like *atpH*, it is thinkable that all three candidates at least partly rely on SD-interactions for initiation that may be supported by uS11c. However, a perfect SD-sequence like in the case of *atpH* might partly compensate for the loss of uS11c. The *psbC* transcript bears a SD-like sequence in its 5'-UTR as well, close to the initiation site. However, its profile exhibits an accumulation of RPFs at the 5'-terminus of the transcript, that together with the occurrence of other translationally important sites within the 5'-UTR suggest a eukaryote-like scanning mechanism for initiation. The mode of initiation and the relevance of SD sequences in the chloroplast has been a topic of ongoing discussion for a long time (Drechsel & Bock, 2011; Fargo et al., 1998; Plader & Sugiura, 2003; Scharff et al., 2017; Wei & Xia, 2019). Together, these results indicate that initiation in the chloroplast is more complex and the importance of SD(-like) sequences depends on the specific transcript to be translated. Especially regarding the small set of transcripts encoded on the chloroplast genome, it appears logical that the translation machinery has evolved very specifically to translating these transcripts with highest fidelity and rate. This would also partly explain the unclear pattern of 5'-P site and 3'-A-site offsets observed for the chloroplast RPFs in this work.

4.7 Selective ribosome profiling identifies *psaB* as target of cpSRP54 and may suggest its involvement in co-translational protein import into the chloroplast

Although the cpSRP54-specific ribosome profiling experiment revealed enriched transcripts, the results raised further questions. Among the 129 enriched candidates, only one (*orf528* – encoding a protein of unknown function) was encoded in the chloroplast but many in the nucleus. Given the localization of cpSRP54 in the chloroplast stroma, it should not be able to interact directly with cytosolic ribosomes and therefore the enrichment of cytosolic RPFs seems counterintuitive. A particularly interesting observation of the experiment was that the transcript encoding cpSRP54 itself was found among the top enriched candidates. While such

a finding would be expected in a proteomics experiment, in case of this seRP experiment it suggests frequent interaction of chloroplast-localized cpSRP54 with cpSRP54-translating cytosolic ribosomes. A possible explanation for this observation may be found in a mechanism of co-translational protein import into the chloroplast that was described recently (Sun et al., 2024). This mechanism was speculated to be facilitated via the TIC-TOC pore complex in the chloroplast envelopes. The observations made in the seRP experiment could be explained via an interaction of cpSRP54 with nascent chains that may be co-translationally threaded through this pore complex into the chloroplast stroma, where cpSRP54 could potentially bind their N-termini, yielding an RPF of the cytosolic ribosome in a seRP-experiment. Such a mechanism would be plausible especially because of its similarity to the generally accepted role of cpSRP54 in post-translational import of LHC-proteins into the chloroplast (Ziehe et al., 2018). In this post-translational import mechanism, the protein to be imported is translated within the cytosol and is kept in an unfolded state by chaperones which accompany the protein onto the TIC-TOC pore complex for import into the stroma. At the stromal side of the inner envelope, cpSRP54 awaits the polypeptide chain and after binding it together with SRP43, directs it to the thylakoid membrane (Goforth et al., 2004; Schuenemann et al., 1998; Yuan et al., 2002). cpSRP54's general mode of action – which is binding to an unfolded polypeptide chain that is threaded through the TIC-TOC pore complex – would be the same, regardless of post- or co-translational import. Interestingly, not a single LHC-encoding transcript was found to be enriched in the seRP experiment, indicating that a putative involvement of cpRSP54 in co-translational protein import would not apply to LHC proteins. However, the results of this experiment alone are not sufficient to prove that cpSRP54 has a role in co-translational protein import. Hence, further experiments with strict controls are necessary to verify these indicators towards a putatively undiscovered import pathway.

Only PsaB and FtsH were identified as targets of cpSRP54's co-translational action towards chloroplast ribosomes. FtsH is a thylakoid membrane bound protease, which was shown to be involved in the degradation of chloroplast membrane proteins in higher plants (Kato et al., 2023). It was observed that FtsH's proteolytic activity towards LHC proteins dramatically decreases in knockout mutants of cpSRP54, also indicating that the protein relies on co-translational membrane insertion facilitated by cpSRP54 (Lei et al., 2023).

In higher plants, photosystem II core subunit D1 was identified as a main target of cpSRp54's co-translational action (Nilsson & van Wijk, 2002). In contrast to this, only weak local enrichments of *psbA* were found in this study, indicating that the strength or duration of cpSRP54's interaction with nascent chains may differ and that interaction with nascent chains during *psbA* translation may be especially fragile or transient.

Despite the necessity of follow-up experiments, some findings about cpSRP54's co-translational role regarding chloroplast-encoded proteins could be drawn from the data. Through a strong and sharp local enrichment of RPFs encoding codons 105 to 120 of *psaB*, the PS I subunit was identified as one of two co-translational substrate candidates of cpSRP54 in the *C. reinhardtii* chloroplast. Generally, cpSRP54 is assumed to direct candidates for thylakoid insertion onto the thylakoid membrane (Ziehe et al., 2018). The identified point of cpSRP54-ribosome interaction is in line with this assumption. After synthesis, the polypeptide chain has to trespass the polypeptide exit tunnel which can harbor a chain of approximately 30 to 40 amino acids (Chadani et al., 2021). Due to this offset, the first transmembrane domain of PsaB (codon 47 – 70) emerges from the ribosome when cpSRP54 is binding during translation of codons 105 to 120 (assuming a 40 amino acid offset). The emergence of the first transmembrane domain also represents the earliest possible time point for insertion into the thylakoid membrane. To verify and understand this interaction, further experiments are required including mutagenesis of PsaB's first transmembrane domain to identify the exact binding site and KO-mutagenesis of cpSRP54 to specifically investigate the consequences regarding psaB translation, folding and membrane insertion.

The virtual absence of globally enriched chloroplast-encoded transcripts in the seRP experiment indicates that cpSRP54 transiently interacts with active ribosomes upon its co-translational action. Instead of generally enriched candidates, sharp peaks were found within the specific enrichment profiles of *psaB* and *ftsH*. These findings contrast proposed binding modes of the higher plant cpSRP54 that suggest a binding of the protein to ribosomes even before the nascent chain emerges from the polypeptide channel (Hristou et al., 2019). The results of the seRP experiment and especially the enrichment profiles of chloroplast transcripts instead support a mode in which cpSRP54 in *C. reinhardtii* only transiently binds to specific regions of the nascent chain. The fact that these enriched regions were very short in both cases observed in this study indicates that translation is presumably paused upon binding of cpSRP54 and only resumes after the translational co-factor has dissociated from the nascent chain.

4.8 Outlook

The results of this work shed light on the translatome of *C. reinhardtii* and revealed numerous details not only regarding the expression of transcripts but also on the dynamics of translation. However, although the experiments presented in this study yielded a great number of insights and sparked multiple hypotheses, the data also raised more questions and calls for validation by orthogonal methods and deeper analyses.

The analysis of the 150 top-translated transcripts revealed several yet uncharacterized candidates which according to their expression strength appear to play an important role in cells under mixotrophic conditions. These transcripts represent interesting candidates for future studies to investigate their detailed role in the cell. In general, a comparison of mixotrophic against photoautotrophic cultures might yield interesting insights and could answer the question of how much mixotrophic cultures reflect the physiological state of *C. reinhardtii* in its natural habitat where external carbon sources are presumably much less abundant. Moreover, the cultivation under mixotrophic conditions with permanent illumination may mask dependencies and co-regulations between different transcripts due to a non-synchronous cell cycle between cells in the culture. It would be interesting to cultivate cells photoautotrophically in day-night cycles to achieve synchronization of the cell cycle in the cultures and inspect the translatome at multiple sampling points. Such an experiment would be helpful not only to find transcripts co-regulated at the translational level under physiological conditions but comparison of ribosome profiles at different sampling points could also reflect how translation dynamics of individual transcripts change over the diurnal cycle and may shed light on strategies of translational regulation.

A very general question regarding the interpretation of a single-point Ribo-Seq measurement is how much it truly reflects a protein synthesis rate. While the number of RPFs mapping to a transcript reflects the number of ribosomes that were bound to that transcript species, pausing events or differing elongation speeds across transcripts may have an impact on how “productive” these ribosomes truly are. A true protein synthesis rate could be determined by time-resolved Ribo-Seq kinetics in cultures treated with translation initiation inhibitors (e.g. harringtonine for cytosolic ribosomes and lincomycin for organellar ribosomes). In such samples, the ribosome density of transcripts should decrease quickly over time, and the loss of density should correlate with the rate of protein synthesis for every transcript individually. Also, a comparison of such data as of a single-point Ribo-Seq measurement with pcSILAC as an orthogonal approach could be helpful to assess the accuracy of Ribo-Seq expression data.

The data produced in this study could be utilized to systematically assess genomic annotations in *C. reinhardtii*, especially regarding falsely annotated start codons, by comparing the

ribosome profiles of all transcripts against the current annotation. Such a comparison could help refining the genome and could possibly be able to map upstream ORFs and events of near-cognate translation initiation.

In this study four transcripts were reported to be translationally dependent on the chloroplast ribosomal protein uS11c. It was hypothesized that for three of them, Shine-Dalgarno(-like) sequences may control this dependency. This hypothesis could be easily tested in a series of follow-up experiments by introducing a reporter gene coupled to 5'-UTRs of affected and non-affected transcripts into the mutant and subsequent mutation of the SD(-like) sequences in these 5'-UTRs. Using Ribo-Seq in induced and non-induced state of the mutant as a read-out, the reporter should be affected by uS11c depletion in the same way as the transcript the respective 5'-UTR was lent from except if the SD(-like) sequence is mutated.

The seRP experiment presented in this study yielded a long list of nuclear encoded transcripts that were enriched in the cpSRP54 pulldown. It was hypothesized that such an observation might be expectable if the protein was involved in a recently proposed co-translational protein import pathway into the chloroplast. However, this unexpected result also requires further investigation. It would be helpful to test the seRP protocol using a chloroplast ribosomal protein as target. In such a case, only enrichment of chloroplast transcripts should be possible, allowing to assess if the nuclear encoded transcripts enriched in the cpSRP54 pulldown may represent an artifact of the method. Furthermore, the cpSRP54 antibody should be carefully examined for cross reactivity with non-target protein, for example by mass spectrometric measurements of protein pulldowns performed with this antibody. If the results of such tests do not indicate that the enrichment of nuclear transcripts is an artifact, the matter should be investigated in greater depth. This could be done by another selective ribosome profiling experiment after pre-incubation with a cytosolic initiation inhibitor or attempts to block the TOC-TIC pore complex. In case no enrichment of nuclear transcripts would be detectable in such experiments anymore, this would strongly support the hypothesis of an involvement of cpSRP54 in co-translational protein import into the chloroplast.

5 Summary

Translation is a ubiquitous, yet enigmatic process central to all cellular life on earth. Despite being a subject of research for many decades, its enormous complexity makes it difficult to study translation. The development of the Ribo-Seq technique, a combination of classical ribosome profiling and next generation sequencing, enabled scientists to get a closer view at translating ribosomes than ever before. This technique allows to investigate the topic with unprecedented detail and to generate deep insights into the mechanics and regulation of translation. In this study, a Ribo-Seq protocol for *C. reinhardtii* and a Python module for the analysis of Ribo-Seq data was presented. The data generated was analyzed translatome-wide and its quality was assessed regarding sequencing depth, the RPF's frame preference and their length distribution. The data delivered deep insights into the mixotrophic translatome, clearly demonstrating the dominance of anabolism-related transcripts. Analysis of 5'-P-site offsets shed light on the mechanics of cytosolic translation revealing a clear tri-nucleotide periodicity of cytosolic RPFs. It was shown that ribosome profiles reproducibly reflect the dynamics of translation and can be utilized to refine genomic annotations. The appearance of an RNAi-related ribosome-binding protein (VIG1) among the 150 top-translated transcripts, together with the frequent observation of extreme RPF-coverage within initiation regions indicates that regulation of cytosolic translation in *C. reinhardtii* may be more complex than previously anticipated. Analysis of chloroplast RPFs indicated that chloroplasts evolved highly specific mechanisms to fine-tune translation and that translational activity in the chloroplast may be regulated on a high level by the titers or ribosomal proteins Rpl20 and PRPL7. An analysis of the translatome of a knock-down mutant of uS11c revealed four transcripts depending on the protein for translation as well as a list of transcripts putatively involved in the formation of palmelloid colonies. Additionally, a seRP approach suggested *psaB* and *ftsH* as targets of a transient co-translational interaction of cpSRP54 with the chloroplast ribosome and pointed towards the protein's involvement in a putative co-translational protein import pathway into the chloroplast. Parts of this study have been published previously in a scientific Journal in the publication "Utilizing high-resolution ribosome profiling for the global investigation of gene expression in *Chlamydomonas*" (Gotsmann et al., 2024). Furthermore, the RPF-coverage of genes in the *C. reinhardtii* genome, version 6.1 was published on the Phytozome platform (Goodstein et al., 2012) to enable the scientific community to compare existing genomic annotations against Ribo-Seq data.

6 Zusammenfassung

Die Translation ist ein allgegenwärtiger und doch rätselhafter Prozess, der von zentraler Bedeutung für alles zelluläre Leben ist. Obwohl dieser Prozess bereits seit Jahrzehnten Forschungsgegenstand ist, gestaltet sich seine Untersuchung aufgrund seiner Komplexität schwierig. Die Entwicklung von Ribo-Seq, eine Kombination aus Ribosome-Profiling und Next Generation Sequencing, ermöglichte es Forschenden, translatierende Ribosomen mit größter Detailtiefe zu untersuchen und tiefgreifende Erkenntnisse bezüglich der Mechanik und Regulation der Translation zu gewinnen. In dieser Studie wurden ein Ribo-Seq Protokoll für *C. reinhardtii* sowie ein Python Modul für die Auswertung von Ribo-Seq Daten vorgestellt. Die erzeugten Daten wurden Translatom-weit ausgewertet und ihre Qualität wurde in Hinblick auf Sequenziertiefe sowie Leserahmen-Präferenz und Längenverteilung der extrahierten RPFs beurteilt. Die Daten lieferten tiefe Einblicke in das mixotrophe Translatom und konnten klar die Dominanz Anabolismus-verwandter Transkripte in diesem aufzeigen. Die Auswertung der 5‘-P-site Abstände ermöglichte zudem Einsicht in die Mechanik der Translation und zeigte eine deutliche tri-Nukleotid Periodizität cytosolischer RPFs. Es wurde gezeigt, dass Ribosomen-Profile reproduzierbar die Dynamik der Translation widerspiegeln und zur Verbesserung genomischer Annotationen genutzt werden können. Die Anwesenheit eines RNAi-assoziierten, Ribosomen-bindenden Proteins (VIG1) unter den 150 am stärksten translatierten Transkripten deutete zusammen mit der häufigen Beobachtung extremer Akkumulationen von RPFs im Bereich von Startcodonen darauf hin, dass die Regulation der cytosolischen Translation in *C. reinhardtii* deutlich komplexer sein könnte als bisher angenommen. Die Analyse chloroplastidärer RPFs wies darauf hin, dass Chloroplasten hochspezifische Mechanismen zur Feinjustierung der Translation entwickelt haben und dass die Aktivität der Translation im Chloroplasten auf übergeordneter Ebene durch die Titer der ribosomalen Proteine Rpl20 und PRPL7 reguliert sein könnte. Die Analyse des Translatoms einer knock-down Mutante von uS11c konnte vier Transkripte identifizieren, deren Translation direkt von dem Protein abhängt sowie eine Liste von Transkripten liefern, welche vermutlich eine Rolle bei der Bildung von Palmelloid-Kolonien spielen. Zusätzlich wurden PsaB und FtsH in einem seRP-Ansatz als Ziele einer transienten, kotranslationalen Interaktion von cpSRP54 mit dem chloroplastidären Ribosom vorgeschlagen sowie Hinweise auf eine mögliche Beteiligung des Proteins an einem mutmaßlich kotranslationalen Proteinimportweg in den Chloroplasten entdeckt. Teile dieser Arbeit wurden bereits in einem wissenschaftlichen Journal in der Publikation „Utilizing high-resolution ribosome profiling for the global investigation of gene expression in *Chlamydomonas*“ (Gotsmann et al., 2024) veröffentlicht. Des Weiteren wurde auch die RPF-Abdeckung von Genen im *C. reinhardtii* Genom, Version 6.1 auf der Plattform Phytozome (Goodstein et al., 2012) veröffentlicht, um der wissenschaftlichen Gemeinschaft die Überprüfung vorhandener Genannotationen anhand von Ribo-Seq Daten zu ermöglichen.

7 List of cited literature

Afgan, E., Nekrutenko, A., Grüning, B. A., Blankenberg, D., Goecks, J., Schatz, M. C., Ostrovsky, A. E., Mahmoud, A., Lonie, A. J., Syme, A., Fouilloux, A., Bretaudeau, A., Nekrutenko, A., Kumar, A., Eschenlauer, A. C., Desanto, A. D., Guerler, A., Serrano-Solano, B., Batut, B., … Briggs, P. J. (2022). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. *Nucleic Acids Research*, *50*(W1), W345–W351. https://doi.org/10.1093/NAR/GKAC247

Almagro Armenteros, J. J., Salvatore, M., Emanuelsson, O., Winther, O., von Heijne, G., Elofsson, A., & Nielsen, H. (2019). Detecting sequence signals in targeting peptides using deep learning. *Life Science Alliance*, *2*(5), e201900429. https://doi.org/10.26508/lsa.201900429

Amin, M. R., Yurovsky, A., Chen, Y., Skiena, S., & Futcher, B. (2018). Re-annotation of 12,495 prokaryotic 16S rRNA 3' ends and analysis of Shine-Dalgarno and anti-Shine-Dalgarno sequences. *PLOS ONE*, *13*(8), e0202767-. https://doi.org/10.1371/journal.pone.0202767

Andrews, S., Krueger, F., Segonds-Pichon, A., Biggins, L., Krueger, C., & Wingett, S. (2012). *FastQC*.

Anfinsen, C. B. (1973). Principles that Govern the Folding of Protein Chains. *Science*, *181*(4096), 223–230. https://doi.org/10.1126/science.181.4096.223

Aro, E.-M., Virgin, I., & Andersson, B. (1993). Photoinhibition of Photosystem II. Inactivation, protein damage and turnover. *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, *1143*(2), 113–134. https://doi.org/https://doi.org/10.1016/0005-2728(93)90134-2

Babitzke, P., Baker, C. S., & Romeo, T. (2009). Regulation of translation initiation by RNA binding proteins. In *Annual Review of Microbiology* (Vol. 63, pp. 27–44). https://doi.org/10.1146/annurev.micro.091208.073514

Baek, D., Villén, J., Shin, C., Camargo, F. D., Gygi, S. P., & Bartel, D. P. (2008). The impact of microRNAs on protein output. *Nature*, *455*(7209), 64–71. https://doi.org/10.1038/nature07242

Ban, N., Beckmann, R., Cate, J. H. D., Dinman, J. D., Dragon, F., Ellis, S. R., Lafontaine, D. L. J., Lindahl, L., Liljas, A., Lipton, J. M., McAlear, M. A., Moore, P. B., Noller, H. F., Ortega, J., Panse, V. G., Ramakrishnan, V., Spahn, C. M. T., Steitz, T. A., Tchorzewski, M., … Yusupov, M. (2014). A new system for naming ribosomal proteins. *Current Opinion in Structural Biology*, *24*, 165–169. https://doi.org/https://doi.org/10.1016/j.sbi.2014.01.002

Becker, A. H., Oh, E., Weissman, J. S., Kramer, G., & Bukau, B. (2013). Selective ribosome profiling as a tool for studying the interaction of chaperones and targeting factors with nascent polypeptide chains and ribosomes. *Nature Protocols*, *8*(11), 2212–2239. https://doi.org/10.1038/nprot.2013.133

Beligni, M. V., Yamaguchi, K., & Mayfield, S. P. (2004). Chloroplast Elongation Factor Ts Pro-Protein Is an Evolutionarily Conserved Fusion with the S1 Domain-Containing Plastid-Specific Ribosomal Protein-7. *The Plant Cell*, *16*(12), 3357–3369. https://doi.org/10.1105/tpc.104.026708

Bendich, A. J., & McCarthy, B. J. (1970). Ribosomal RNA Homologies among Distantly Related Organisms. *Proceedings of the National Academy of Sciences*, *65*(2), 349–356. https://doi.org/10.1073/pnas.65.2.349

Benjamini, Y., & Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, *57*(1), 289–300. http://www.jstor.org/stable/2346101

Bertolini, M., Fenzl, K., Kats, I., Wruck, F., Tippmann, F., Schmitt, J., Auburger, J. J., Tans, S., Bukau, B., & Kramer, G. (2021). Interactions between nascent proteins translated by adjacent ribosomes drive homomer assembly. *Science*, *371*(6524), 57–64. https://doi.org/10.1126/science.abc7151

Bieri, P., Leibundgut, M., Saurer, M., Boehringer, D., & Ban, N. (2017). The complete structure of the chloroplast 70S ribosome in complex with translation factor pY. *The EMBO Journal*, *36*(4), 475–486. https://doi.org/10.15252/embj.201695959

Bonaventure, G., & Ohlrogge, J. B. (2002). Differential Regulation of mRNA Levels of Acyl Carrier Protein Isoforms in Arabidopsis. *Plant Physiology*, *128*(1), 223–235. https://doi.org/10.1104/pp.010397

Bostrom, K., Wettesten, M., Boren, J., Bondjers, G., Wiklund, O., & Olofsson, S. O. (1986). Pulse-chase studies of the synthesis and intracellular transport of apolipoprotein B-100 in Hep G2 cells. *Journal of Biological Chemistry*, *261*(29), 13800–13806. https://doi.org/10.1016/s0021-9258(18)67090-5

Boudreau, E., Takahashi, Y., Lemieux, C., Turmel, M., & Rochaix, J. (1997). The chloroplast ycf3 and ycf4 open reading frames of Chlamydomonas reinhardtii are required for the accumulation of the photosystem I complex. *The EMBO Journal*, *16*(20), 6095-6104–6104. https://doi.org/https://doi.org/10.1093/emboj/16.20.6095

Bowman, J. C., Petrov, A. S., Frenkel-Pinter, M., Penev, P. I., & Williams, L. D. (2020). Root of the Tree: The Significance, Evolution, and Origins of the Ribosome. In *Chemical Reviews* (Vol. 120, Issue 11, pp. 4848–4878). American Chemical Society. https://doi.org/10.1021/acs.chemrev.9b00742

Breindel, L., Yu, J., Burz, D. S., & Shekhtman, A. (2020). Intact ribosomes drive the formation of protein quinary structure. *PLOS ONE*, *15*(4), e0232015-. https://doi.org/10.1371/journal.pone.0232015

Brilot, A. F., Korostelev, A. A., Ermolenko, D. N., & Grigorieff, N. (2013). Structure of the ribosome with elongation factor G trapped in the pretranslocation state. *Proceedings of the National Academy of Sciences*, *110*(52), 20994–20999. https://doi.org/10.1073/pnas.1311423110

Budkevich, T., Giesebrecht, J., Altman, R. B., Munro, J. B., Mielke, T., Nierhaus, K. H., Blanchard, S. C., & Spahn, C. M. T. (2011). Structure and Dynamics of the Mammalian Ribosomal Pretranslocation Complex. *Molecular Cell*, *44*(2), 214–224. https://doi.org/10.1016/j.molcel.2011.07.040

Chadani, Y., Sugata, N., Niwa, T., Ito, Y., Iwasaki, S., & Taguchi, H. (2021). Nascent polypeptide within the exit tunnel stabilizes the ribosome to counteract risky translation. *The EMBO Journal*, *40*(23), e108299. https://doi.org/https://doi.org/10.15252/embj.2021108299

Chambers, C. M., & Ness, G. C. (1997). Translational Regulation of Hepatic HMG-CoA Reductase by Dietary Cholesterol. *Biochemical and Biophysical Research Communications*, *232*(2), 278–281. https://doi.org/https://doi.org/10.1006/bbrc.1997.6288

Chassé, H., Boulben, S., Costache, V., Cormier, P., & Morales, J. (2017). Analysis of translation using polysome profiling. *Nucleic Acids Research*, *45*(3), e15–e15. https://doi.org/10.1093/nar/gkw907

Chaux, F., Jarrige, D., Rodrigues-Azevedo, M., Bujaldon, S., Caspari, O. D., Ozawa, S.-I., Drapier, D., Vallon, O., Choquet, Y., & de Vitry, C. (2023). Chloroplast ATP synthase biogenesis requires peripheral stalk subunits AtpF and ATPG and stabilization of atpE mRNA by OPR protein MDE1. *The Plant Journal*, *116*(6), 1582–1599. https://doi.org/https://doi.org/10.1111/tpj.16448

Chen, H., Bjerknes, M., Kumar, R., & Jay, E. (1994). Determination of the optimal aligned spacing between the Shine – Dalgarno sequence and the translation initiation codon of Escherichia coli m RNAs. *Nucleic Acids Research*, *22*(23), 4953–4957. https://doi.org/10.1093/nar/22.23.4953

Chen, X., Wei, S., Ji, Y., Guo, X., & Yang, F. (2015). Quantitative proteomics using SILAC: Principles, applications, and developments. *PROTEOMICS*, *15*(18), 3175–3192. https://doi.org/https://doi.org/10.1002/pmic.201500108

Chitnis, P. R., & Thornber, J. P. (1988). The major light-harvesting complex of Photosystem II: aspects of its molecular and cell biology. *Photosynthesis Research*, *16*(1), 41–63. https://doi.org/10.1007/BF00039485

Chotewutmontri, P., & Barkan, A. (2018). Multilevel effects of light on ribosome dynamics in chloroplasts program genome-wide and psbA-specific changes in translation. *PLOS Genetics*, *14*(8), e1007555-. https://doi.org/10.1371/journal.pgen.1007555

Chung, B. Y., Hardcastle, T. J., Jones, J. D., Irigoyen, N., Firth, A. E., Baulcombe, D. C., & Brierley, I. (2015). The use of duplex-specific nuclease in ribosome profiling and a user-friendly software package for Ribo-seq data analysis. *RNA*, *21*(10), 1731–1745. https://doi.org/10.1261/rna.052548.115

Chung, B. Y. W., Deery, M. J., Groen, A. J., Howard, J., & Baulcombe, D. C. (2017). Endogenous miRNA in the green alga Chlamydomonas regulates gene expression through CDS-targeting. *Nature Plants*, *3*(10), 787. https://doi.org/10.1038/S41477-017-0024-6

Cleveland, W. S. (1981). LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression. *The American Statistician*, *35*(1), 54. https://doi.org/10.2307/2683591

Cohen, R. D., & Pielak, G. J. (2017). A cell is more than the sum of its (dilute) parts: A brief history of quinary structure. *Protein Science*, *26*(3), 403–413. https://doi.org/https://doi.org/10.1002/pro.3092

Craig, R. J., Gallaher, S. D., Shu, S., Salomé, P. A., Jenkins, J. W., Blaby-Haas, C. E., Purvine, S. O., O'Donnell, S., Barry, K., Grimwood, J., Strenkert, D., Kropat, J., Daum, C., Yoshinaga, Y., Goodstein, D. M., Vallon, O., Schmutz, J., & Merchant, S. S. (2023). The Chlamydomonas Genome Project, version 6: Reference assemblies for mating-type plus and minus strains reveal extensive structural mutation in the laboratory. *The Plant Cell*, *35*(2), 644–672. https://doi.org/10.1093/plcell/koac347

Crick, F. (1970). Central Dogma of Molecular Biology. *Nature*, *227*(5258), 561–563. https://doi.org/10.1038/227561a0

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., & Li, H. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2), giab008. https://doi.org/10.1093/gigascience/giab008

Davydov, I. I., Wohlgemuth, I., Artamonova, I. I., Urlaub, H., Tonevitsky, A. G., & Rodnina, M. V. (2013). Evolution of the protein stoichiometry in the L12 stalk of bacterial and organellar ribosomes. *Nature Communications*, *4*(1), 1387. https://doi.org/10.1038/ncomms2373

de Vitry, C., Olive, J., Drapier, D., Recouvreur, M., & Wollman, F. A. (1989). Posttranslational events leading to the assembly of photosystem II protein complex: a study using photosynthesis mutants from Chlamydomonas reinhardtii. *Journal of Cell Biology*, *109*(3), 991–1006. https://doi.org/10.1083/jcb.109.3.991

DeMott, C. M., Majumder, S., Burz, D. S., Reverdatto, S., & Shekhtman, A. (2017). Ribosome Mediated Quinary Interactions Modulate In-Cell Protein Activities. *Biochemistry*, *56*(32), 4117–4126. https://doi.org/10.1021/acs.biochem.7b00613

Denovan-Wright, E. M., Nedelcu, A. M., & Lee, R. W. (1998). Complete sequence of the mitochondrial DNA of Chlamydomonas eugametos. *Plant Molecular Biology*, *36*(2), 285–295. https://doi.org/10.1023/A:1005995718091

Dever, T. E., Dinman, J. D., & Green, R. (2018). Translation Elongation and Recoding in Eukaryotes. *Cold Spring Harbor Perspectives in Biology*, *10*(8), a032649. https://doi.org/10.1101/cshperspect.a032649

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., & Gingeras, T. R. (2013). STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, *29*(1), 15–21. https://doi.org/10.1093/bioinformatics/bts635

Dobin, A., & Gingeras, T. R. (2015). Mapping RNA-seq Reads with STAR. *Current Protocols in Bioinformatics / Editoral Board, Andreas D. Baxevanis ... [et Al.]*, *51*(1), 11.14.1. https://doi.org/10.1002/0471250953.BI1114S51

Doerfel, L. K., Wohlgemuth, I., Kubyshkin, V., Starosta, A. L., Wilson, D. N., Budisa, N., & Rodnina, M. V. (2015). Entropic Contribution of Elongation Factor P to Proline Positioning at the Catalytic Center of the Ribosome. *Journal of the American Chemical Society*, *137*(40), 12997–13006. https://doi.org/10.1021/jacs.5b07427

Dong, L., Mao, Y., Zhou, A., Liu, X. M., Zhou, J., Wan, J., & Qian, S. B. (2021). Relaxed initiation pausing of ribosomes drives oncogenic translation. *Science Advances*, *7*(8). https://doi.org/10.1126/SCIADV.ABD6927/SUPPL_FILE/ABD6927_SM.PDF

Drechsel, O., & Bock, R. (2011). Selection of Shine-Dalgarno sequences in plastids. *Nucleic Acids Research*, *39*(4), 1427–1438. https://doi.org/10.1093/nar/gkq978

Dunn, J. G., & Weissman, J. S. (2016). Plastid: nucleotide-resolution analysis of next-generation sequencing and genomics data. *BMC Genomics*, *17*(1), 958. https://doi.org/10.1186/s12864-016-3278-x

Durante, L., Hübner, W., Lauersen, K. J., & Remacle, C. (2019). Characterization of the GPR1/FUN34/YaaH protein family in the green microalga Chlamydomonas suggests their role as intracellular membrane acetate channels. *Plant Direct*, *3*(6), e00148. https://doi.org/https://doi.org/10.1002/pld3.148

Dutcher, S. K. (1995). Mating and Tetrad Analysis in Chlamydomonas reinhardtii. In W. Dentler & G. Witman (Eds.), *Methods in Cell Biology* (Vol. 47, pp. 531–540). Academic Press. https://doi.org/https://doi.org/10.1016/S0091-679X(08)60857-2

Dyer, S. C., Austine-Orimoloye, O., Azov, A. G., Barba, M., Barnes, I., Barrera-Enriquez, V. P., Becker, A., Bennett, R., Beracochea, M., Berry, A., Bhai, J., Bhurji, S. K., Boddu, S., Branco Lins, P. R., Brooks, L., Ramaraju, S. B., Campbell, L. I., Martinez, M. C., Charkhchi, M., … Yates, A. D. (2025). Ensembl 2025. *Nucleic Acids Research*, *53*(D1), D948–D957. https://doi.org/10.1093/nar/gkae1071

Esquerré, T., Laguerre, S., Turlan, C., Carpousis, A. J., Girbal, L., & Cocaign-Bousquet, M. (2014). Dual role of transcription and transcript stability in the regulation of gene expression in Escherichia coli cells cultured on glucose at different growth rates. *Nucleic Acids Research*, *42*(4), 2460–2472. https://doi.org/10.1093/nar/gkt1150

Estrada, K., Garciarrubio, A., & Merino, E. (2024). Unraveling the plasticity of translation initiation in prokaryotes: Beyond the invariant Shine-Dalgarno sequence. *PLOS ONE*, *19*(1), e0289914-. https://doi.org/10.1371/journal.pone.0289914

Eulalio, A., Huntzinger, E., & Izaurralde, E. (2008). GW182 interaction with Argonaute is essential for miRNA-mediated translational repression and mRNA decay. *Nature Structural & Molecular Biology*, *15*(4), 346–353. https://doi.org/10.1038/nsmb.1405

Fabbretti, A., Pon, C. L., Hennelly, S. P., Hill, W. E., Lodmell, J. S., & Gualerzi, C. O. (2007). The Real-Time Path of Translation Factor IF3 onto and off the Ribosome. *Molecular Cell*, *25*(2), 285–296. https://doi.org/10.1016/j.molcel.2006.12.011

Fargo, D. C., Zhang, M., Gillham, N. W., & Boynton, J. E. (1998). Shine-Dalgarno-like sequences are not required for translation of chloroplast mRNAs in Chlamydomonas reinhardtii chloroplasts or in Escherichia coli. *Molecular and General Genetics MGG*, *257*(3), 271–282. https://doi.org/10.1007/s004380050648

Fellerer, C., Schweiger, R., Schöngruber, K., Soll, J., & Schwenkert, S. (2011). Cytosolic HSP90 Cochaperones HOP and FKBP Interact with Freshly Synthesized Chloroplast Preproteins of Arabidopsis. *Molecular Plant*, *4*(6), 1133–1145. https://doi.org/10.1093/mp/ssr037

Fierro-Monti, I., Racle, J., Hernandez, C., Waridel, P., Hatzimanikatis, V., & Quadroni, M. (2013). A Novel Pulse-Chase SILAC Strategy Measures Changes in Protein Decay and Synthesis Rates Induced by Perturbation of Proteostasis with an Hsp90 Inhibitor. *PLOS ONE*, *8*(11), e80423-. https://doi.org/10.1371/journal.pone.0080423

Findinier, J., Joubert, L.-M., Fakhimi, N., Schmid, M. F., Malkovskiy, A. V, Chiu, W., Burlacot, A., & Grossman, A. R. (2024). Dramatic changes in mitochondrial subcellular location and morphology accompany activation of the CO2 concentrating mechanism. *Proceedings of the National Academy of Sciences*, *121*(43), e2407548121. https://doi.org/10.1073/pnas.2407548121

Flanagan, K., Baradaran-Heravi, A., Yin, Q., Dao Duc, K., Spradling, A. C., & Greenblatt, E. J. (2022). FMRP-dependent production of large dosage-sensitive proteins is highly conserved. *Genetics*, *221*(4), iyac094. https://doi.org/10.1093/genetics/iyac094

Fritzsche, S., & Springer, S. (2014). Pulse-chase analysis for studying protein synthesis and maturation. *Current Protocols in Protein Science*, *2014*, 30.3.1-30.3.23. https://doi.org/10.1002/0471140864.ps3003s78

Gallaher, S. D., Craig, R. J., Ganesan, I., Purvine, S. O., McCorkle, S. R., Grimwood, J., Strenkert, D., Davidi, L., Roth, M. S., Jeffers, T. L., Lipton, M. S., Niyogi, K. K., Schmutz, J., Theg, S. M., Blaby-Haas, C. E., & Merchant, S. S. (2021). Widespread polycistronic gene expression in green algae. *Proceedings of the National Academy of Sciences*, *118*(7), e2017714118. https://doi.org/10.1073/pnas.2017714118

Gallaher, S. D., Fitz-Gibbon, S. T., Strenkert, D., Purvine, S. O., Pellegrini, M., & Merchant, S. S. (2018). High-throughput sequencing of the chloroplast and mitochondrion of Chlamydomonas reinhardtii to generate improved de novo assemblies, analyze expression patterns and transcript speciation, and evaluate diversity among laboratory strains and wild isolates. *The Plant Journal*, *93*(3), 545–565. https://doi.org/https://doi.org/10.1111/tpj.13788

Gao, H., Yun-jiao, Z., Yun-chuan, Z., Chang-gui, Q., Ji-hui, L., Jing-jin, Y., Yi-kun, H., Shi-hua, Y., Jing, L., Ze, L., & and Hou, Y. (2016). Complete chloroplast genome sequence of Nicotiana tabacum TN90 (Solanaceae). *Mitochondrial DNA Part B*, *1*(1), 867–868. https://doi.org/10.1080/23802359.2015.1137808

Gawroński, P., Jensen, P. E., Karpiński, S., Leister, D., & Scharff, L. B. (2018). Pausing of Chloroplast Ribosomes Is Induced by Multiple Features and Is Linked to the Assembly of Photosynthetic Complexes . *Plant Physiology*, *176*(3), 2557–2569. https://doi.org/10.1104/pp.17.01564

Geisberg, J. V., Moqtaderi, Z., Fan, X., Ozsolak, F., & Struhl, K. (2014). Global Analysis of mRNA Isoform Half-Lives Reveals Stabilizing and Destabilizing Elements in Yeast. *Cell*, *156*(4), 812–824. https://doi.org/10.1016/j.cell.2013.12.026

Gene Ontology Consortium. (2004). The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, *32*(suppl_1), D258–D261. https://doi.org/10.1093/nar/gkh036

Gerashchenko, M. V, & Gladyshev, V. N. (2017). Ribonuclease selection for ribosome profiling. *Nucleic Acids Research*, *45*(2), e6–e6. https://doi.org/10.1093/nar/gkw822

Gilbert, W. (1986). Origin of life: The RNA world. *Nature*, *319*(6055), 618. https://doi.org/10.1038/319618a0

Glaub, A., Huptas, C., Neuhaus, K., & Ardern, Z. (2020). Recommendations for bacterial ribosome profiling experiments based on bioinformatic evaluation of published data. *Journal of Biological Chemistry*, *295*(27), 8999–9011. https://doi.org/10.1074/jbc.RA119.012161

Goddard, T. D., Huang, C. C., Meng, E. C., Pettersen, E. F., Couch, G. S., Morris, J. H., & Ferrin, T. E. (2018). UCSF ChimeraX: Meeting modern challenges in visualization and analysis. *Protein Science*, *27*(1), 14–25. https://doi.org/https://doi.org/10.1002/pro.3235

Goforth, R. L., Peterson, E. C., Yuan, J., Moore, M. J., Kight, A. D., Lohse, M. B., Sakon, J., & Henry, R. L. (2004). Regulation of the GTPase Cycle in Post-translational Signal Recognition Particle-based Protein Targeting Involves cpSRP43 *. *Journal of Biological Chemistry*, *279*(41), 43077–43084. https://doi.org/10.1074/jbc.M401600200

Goodenough, U. W., & Heuser, J. E. (1985). *The Chlamydomonas Cell Wall Glycoproteins Analyzed by the Technique and Its Constituent Quick-Freeze, Deep-Etch*. *101*(October).

Goodstein, D. M., Shu, S., Howson, R., Neupane, R., Hayes, R. D., Fazo, J., Mitros, T., Dirks, W., Hellsten, U., Putnam, N., & Rokhsar, D. S. (2012). Phytozome: a comparative platform for green plant genomics. *Nucleic Acids Research*, *40*(D1), D1178–D1186. https://doi.org/10.1093/nar/gkr944

Gornicki, P., Nurse, K., Hellmann, W., Boublik, M., & Ofengand, J. (1984). High resolution localization of the tRNA anticodon interaction site on the Escherichia coli 30 S ribosomal subunit. *Journal of Biological Chemistry*, *259*(16), 10493–10498. https://doi.org/10.1016/s0021-9258(18)90990-7

Gotsmann, V. L. (2018). *Characterization of the Plastid Ribosomal Protein S11 and its Influence on Plastid Translation in Chlamydomonas reinhardtii* [Master thesis]. Technical University of Kaiserslautern.

Gotsmann, V. L., Ting, M. K. Y., Haase, N., Rudorf, S., Zoschke, R., & Willmund, F. (2024). Utilizing high-resolution ribosome profiling for the global investigation of gene expression in Chlamydomonas. *The Plant Journal*, *117*(5), 1614–1634. https://doi.org/https://doi.org/10.1111/tpj.16577

Goyal, A., Belardinelli, R., Maracci, C., Milón, P., & Rodnina, M. V. (2015). Directional transition from initiation to elongation in bacterial translation. *Nucleic Acids Research*, *43*(22), 10700–10712. https://doi.org/10.1093/nar/gkv869

Graciet, E., Lebreton, S., Camadro, J.-M., & Gontero, B. (2003). Characterization of native and recombinant A4 glyceraldehyde 3-phosphate dehydrogenase. *European Journal of Biochemistry*, *270*(1), 129–136. https://doi.org/https://doi.org/10.1046/j.1432-1033.2003.03372.x

Greber, B. J., & Ban, N. (2016). Structure and Function of the Mitochondrial Ribosome. *Annual Review of Biochemistry*, *85*, 103–132. https://doi.org/10.1146/annurev-biochem-060815-014343

Gualerzi, C. O., & Pon, C. L. (2015). Initiation of mRNA translation in bacteria: Structural and dynamic aspects. In *Cellular and Molecular Life Sciences* (Vol. 72, Issue 22, pp. 4341–4367). Birkhauser Verlag AG. https://doi.org/10.1007/s00018-015-2010-3

Guerrier-Takada, C., Gardiner, K., Marsh, T., Pace, N., & Altman, S. (1983). The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. *Cell*, *35*(3, Part 2), 849–857. https://doi.org/https://doi.org/10.1016/0092-8674(83)90117-4

Gutierrez, E., Shin, B.-S., Woolstenhulme, C. J., Kim, J.-R., Saini, P., Buskirk, A. R., & Dever, T. E. (2013). eIF5A Promotes Translation of Polyproline Motifs. *Molecular Cell*, *51*(1), 35–45. https://doi.org/10.1016/j.molcel.2013.04.021

Haentjens-Sitri, J., Allemand, F., Springer, M., & Chiaruttini, C. (2008). A Competition Mechanism Regulates the Translation of the Escherichia coli Operon Encoding Ribosomal Proteins L35 and L20. *Journal of Molecular Biology*, *375*(3), 612–625. https://doi.org/https://doi.org/10.1016/j.jmb.2007.10.058

Hahn, A., Vonck, J., Mills, D. J., Meier, T., & Kühlbrandt, W. (2018). Structure, mechanism, and regulation of the chloroplast ATP synthase. *Science*, *360*(6389), eaat4318. https://doi.org/10.1126/science.aat4318

Hamilton, R., Watanabe, C. K., & de Boer, H. A. (1987). Compilation and comparison of the sequence context around the AUG startcodons in Saccharomyces cerevisiae mRNAs. *Nucleic Acids Research*, *15*(8), 3581–3593. https://doi.org/10.1093/nar/15.8.3581

Harris, C. R., Millman, K. J., der Walt, S. J. van, Gommers, R., Virtanen, P., David Cournapeau, Wieser, E., Taylor, J., Sebastian Berg, Smith, N. J., Kern, R., Hoyer, M. P. and S., van Kerkwijk, M. H., Matthew Brett, Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Pierre Gérard-Marchant, … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Harris, E. H. (2001). Chlamydomonas as a model organism. *Annual Review of Plant Biology*, *52*(Volume 52, 2001), 363–406. https://doi.org/https://doi.org/10.1146/annurev.arplant.52.1.363

Harris, J. K., Kelley, S. T., Spiegelman, G. B., & Pace, N. R. (2003). The Genetic Core of the Universal Ancestor. *Genome Research*, *13*(3), 407–412. http://genome.cshlp.org/content/13/3/407.abstract

He, S., Crans, V. L., & Jonikas, M. C. (2023). The pyrenoid: the eukaryotic CO2-concentrating organelle. *The Plant Cell*, *35*(9), 3236–3259. https://doi.org/10.1093/plcell/koad157

Heifetz, P. B., Förster, B., Osmond, C. B., Giles, L. J., & Boynton, J. E. (2000). Effects of Acetate on Facultative Autotrophy inChlamydomonas reinhardtii Assessed by Photosynthetic Measurements and Stable Isotope Analyses1. *Plant Physiology*, *122*(4), 1439–1446. https://doi.org/10.1104/pp.122.4.1439

Hellen, C. U. T. (2018). Translation Termination and Ribosome Recycling in Eukaryotes. *Cold Spring Harbor Perspectives in Biology*, *10*(10), a032656. https://doi.org/10.1101/cshperspect.a032656

Herold, M., & Nierhaus, K. H. (1987). Incorporation of six additional proteins to complete the assembly map of the 50 S subunit from Escherichia coli ribosomes. *Journal of Biological Chemistry*, *262*(18), 8826–8833. https://doi.org/10.1016/s0021-9258(18)47489-3

Hinnebusch, A. G. (2014). The scanning mechanism of eukaryotic translation initiation. In *Annual Review of Biochemistry* (Vol. 83, pp. 779–812). Annual Reviews Inc. https://doi.org/10.1146/annurev-biochem-060713-035802

Hofmann, J. L., Yang, T. S., Sunol, A. M., & Zia, R. N. (2025). Ribosomal L12 stalks recruit elongation factors to speed protein synthesis in Escherichia coli. *Communications Biology*, *8*(1), 940. https://doi.org/10.1038/s42003-025-08366-4

Hristou, A., Gerlach, I., Stolle, D. S., Neumann, J., Bischoff, A., Dünschede, B., Nowaczyk, M. M., Zoschke, R., & Schünemann, D. (2019). Ribosome-Associated Chloroplast SRP54 Enables Efficient Cotranslational Membrane Insertion of Key Photosynthetic Proteins1. *The Plant Cell*, *31*(11), 2734–2750. https://doi.org/10.1105/tpc.19.00169

Hsu, P. Y., Calviello, L., Wu, H.-Y. L., Li, F.-W., Rothfels, C. J., Ohler, U., & Benfey, P. N. (2016). Super-resolution ribosome profiling reveals unannotated translation events in *Arabidopsis*. *Proceedings of the National Academy of Sciences*, *113*(45), E7126–E7135. https://doi.org/10.1073/pnas.1614788113

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Ingolia, N. T., Ghaemmaghami, S., Newman, J. R. S., & Weissman, J. S. (2009). Genome-Wide Analysis in Vivo of Translation with Nucleotide Resolution Using Ribosome Profiling. *Science*, *324*(5924), 218–223. https://doi.org/10.1126/science.1168978

Ingolia, N. T., Lareau, L. F., & Weissman, J. S. (2011). Ribosome Profiling of Mouse Embryonic Stem Cells Reveals the Complexity and Dynamics of Mammalian Proteomes. *Cell*, *147*(4), 789–802. https://doi.org/10.1016/j.cell.2011.10.002

Inoue-Kashino, N., Kashino, Y., & Takahashi, Y. (2011). Psb30 is a photosystem II reaction center subunit and is required for optimal growth in high light in Chlamydomonas reinhardtii. *Journal of Photochemistry and Photobiology B: Biology*, *104*(1), 220–228. https://doi.org/https://doi.org/10.1016/j.jphotobiol.2011.01.024

Iwasaki, S., Kawamata, T., & Tomari, Y. (2009). Drosophila Argonaute1 and Argonaute2 Employ Distinct Mechanisms for Translational Repression. *Molecular Cell*, *34*(1), 58–67. https://doi.org/10.1016/j.molcel.2009.02.010

Jin, Z., Wan, L., Zhang, Y., Li, X., Cao, Y., Liu, H., Fan, S., Cao, D., Wang, Z., Li, X., Pan, J., Dong, M.-Q., Wu, J., & Yan, Z. (2022). Structure of a TOC-TIC supercomplex spanning two chloroplast envelope membranes. *Cell*, *185*(25), 4788-4800.e13. https://doi.org/10.1016/j.cell.2022.10.030

Jiong, M., Allan, C., & Samuel, K. (2002). Correlations between Shine-Dalgarno Sequences and Gene Features Such as Predicted Expression Levels and Operon Structures. *Journal of Bacteriology*, *184*(20), 5733–5745. https://doi.org/10.1128/jb.184.20.5733-5745.2002

Karin, K.-B., Luisa, W., & Burkhard, B. (2014). The Contractile Vacuole as a Key Regulator of Cellular Water Flow in Chlamydomonas reinhardtii. *Eukaryotic Cell*, *13*(11), 1421–1430. https://doi.org/10.1128/ec.00163-14

Kashino, Y., Takahashi, T., Inoue-Kashino, N., Ban, A., Ikeda, Y., Satoh, K., & Sugiura, M. (2007). Ycf12 is a core subunit in the photosystem II complex. *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, *1767*(11), 1269–1275. https://doi.org/https://doi.org/10.1016/j.bbabio.2007.08.008

Kato, Y., Kuroda, H., Ozawa, S.-I., Saito, K., Dogra, V., Scholz, M., Zhang, G., de Vitry, C., Ishikita, H., Kim, C., Hippler, M., Takahashi, Y., & Sakamoto, W. (2023). Characterization of tryptophan oxidation affecting D1 degradation by FtsH in the photosystem II quality control of chloroplasts. *ELife*, *12*, RP88822. https://doi.org/10.7554/eLife.88822

Kellogg, M. K., Miller, S. C., Tikhonova, E. B., & Karamyshev, A. L. (2021). SRPassing Co-translational Targeting: The Role of the Signal Recognition Particle in Protein Targeting and mRNA Protection. *International Journal of Molecular Sciences*, *22*(12). https://doi.org/10.3390/ijms22126284

Kersten, P. J., & Kirk, T. K. (1987). Involvement of a new enzyme, glyoxal oxidase, in extracellular H2O2 production by Phanerochaete chrysosporium. *Journal of Bacteriology*, *169*(5), 2195–2201. https://doi.org/10.1128/jb.169.5.2195-2201.1987

Kim, J., Klein, P. G., & Mullet, J. E. (1991). Ribosomes pause at specific sites during synthesis of membrane-bound chloroplast reaction center protein D1. *Journal of Biological Chemistry*, *266*(23), 14931–14938. https://doi.org/https://doi.org/10.1016/S0021-9258(18)98567-4

Kiniry, S. J., Michel, A. M., & Baranov, P. V. (2020). Computational methods for ribosome profiling data analysis. *Wiley Interdisciplinary Reviews: RNA*, *11*(3), e1577. https://doi.org/10.1002/WRNA.1577

Klaff, P., & Gruissem, W. (1991). Changes in Chloroplast mRNA Stability during Leaf Development. *The Plant Cell*, *3*(5), 517–529. https://doi.org/10.2307/3869357

Klimyuk, V. I., Persello-Cartieaux, F., Havaux, M., Contard-David, P., Schuenemann, D., Meiherhoff, K., Gouet, P., Jones, J. D. G., Hoffman, N. E., & Nussaume, L. (1999). A Chromodomain Protein Encoded by the Arabidopsis CAO Gene Is a Plant-Specific Component of the Chloroplast Signal Recognition Particle Pathway That Is Involved in LHCP Targeting. *The Plant Cell*, *11*(1), 87–99. https://doi.org/10.1105/tpc.11.1.87

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (pp. 87–90).

Koonin, E. V. (2003). Comparative genomics, minimal gene-sets and the last universal common ancestor. *Nature Reviews Microbiology*, *1*(2), 127–136. https://doi.org/10.1038/nrmicro751

Kornberg, H. L., & Krebs, H. A. (1957). Synthesis of Cell Constituents from C2-Units by a Modified Tricarboxylic Acid Cycle. *Nature*, *179*(4568), 988–991. https://doi.org/10.1038/179988a0

Korobeinikova, A. V., Garber, M. B., & Gongadze, G. M. (2012). Ribosomal proteins: Structure, function, and evolution. In *Biochemistry (Moscow)* (Vol. 77, Issue 6, pp. 562–574). Maik Nauka Publishing / Springer SBM. https://doi.org/10.1134/S0006297912060028

Korostelev, A., Trakhanov, S., Asahara, H., Laurberg, M., Lancaster, L., & Noller, H. F. (2007). Interactions and dynamics of the Shine–Dalgarno helix in the 70S ribosome. *Proceedings of the National Academy of Sciences*, *104*(43), 16840–16843. https://doi.org/10.1073/pnas.0707850104

Kozak, M. (1986). Point mutations define a sequence flanking the AUG initiator codon that modulates translation by eukaryotic ribosomes. *Cell*, *44*(2), 283–292. https://doi.org/10.1016/0092-8674(86)90762-2

Kruger, K., Grabowski, P. J., Zaug, A. J., Sands, J., Gottschling, D. E., & Cech, T. R. (1982). Self-splicing RNA: Autoexcision and autocyclization of the ribosomal RNA intervening sequence of tetrahymena. *Cell*, *31*(1), 147–157. https://doi.org/https://doi.org/10.1016/0092-8674(82)90414-7

Künstner, P., Guardiola, A., Takahashi, Y., & Rochaix, J.-D. (1995). A Mutant Strain of Chlamydomonas reinhardtii Lacking the Chloroplast Photosystem II psbI Gene Grows Photoautotrophically. *Journal of Biological Chemistry*, *270*(16), 9651–9654. https://doi.org/10.1074/jbc.270.16.9651

Lareau, L. F., Hite, D. H., Hogan, G. J., & Brown, P. O. (2014). Distinct stages of the translation elongation cycle revealed by sequencing ribosome-protected mRNA fragments. *ELife*, *3*, e01257. https://doi.org/10.7554/eLife.01257

Lauria, F., Tebaldi, T., Bernabò, P., Groen, E. J. N., Gillingwater, T. H., & Viero, G. (2018). riboWaltz: Optimization of ribosome P-site positioning in ribosome profiling data. *PLOS Computational Biology*, *14*(8), e1006169-. https://doi.org/10.1371/journal.pcbi.1006169

Lefebvre-Legendre, L., Merendino, L., Rivier, C., & Goldschmidt-Clermont, M. (2014). On the Complexity of Chloroplast RNA Metabolism: psaA Trans-splicing Can be Bypassed in Chlamydomonas. *Molecular Biology and Evolution*, *31*(10), 2697–2707. https://doi.org/10.1093/molbev/msu215

Legendre, R., Baudin-Baillieu, A., Hatin, I., & Namy, O. (2015). RiboTools: a Galaxy toolbox for qualitative ribosome profiling analysis. *Bioinformatics (Oxford, England)*, *31*(15), 2586–2588. https://doi.org/10.1093/BIOINFORMATICS/BTV174

Lei, Y., Li, B., Wang, X., Wei, J., Wang, P., Zhao, J., Yu, F., & Qi, Y. (2023). Chloroplast SRP54 and FtsH protease coordinate thylakoid membrane-associated proteostasis in Arabidopsis. *Plant Physiology*, *192*(3), 2318–2335. https://doi.org/10.1093/plphys/kiad199

Li, Y., Wang, X., Li, C., Hu, S., Yu, J., & Song, S. (2014). Transcriptome-wide N6-methyladenosine profiling of rice callus and leaf reveals the presence of tissue-specific competitors involved in selective mRNA modification. *RNA Biology*, *11*(9), 1180–1188. https://doi.org/10.4161/RNA.36281

Ling, C., & Ermolenko, D. N. (2015). Initiation factor 2 stabilizes the ribosome in a semirotated conformation. *Proceedings of the National Academy of Sciences*, *112*(52), 15874–15879. https://doi.org/10.1073/pnas.1520337112

Ling, C., & Ermolenko, D. N. (2016). Structural insights into ribosome translocation. *WIREs RNA*, *7*(5), 620–636. https://doi.org/https://doi.org/10.1002/wrna.1354

Londei, P. (2020). Archaeal Ribosomes. In *eLS* (pp. 1–5). https://doi.org/https://doi.org/10.1002/9780470015902.a0000293.pub3

Londei, P., & Ferreira-Cerca, S. (2021). Ribosome Biogenesis in Archaea. *Frontiers in Microbiology*, *12*. https://www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2021.686977

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, *15*(12), 550. https://doi.org/10.1186/s13059-014-0550-8

Ludwig, W., & Schleifer, K. H. (1994). Bacterial phylogeny based on 16S and 23S rRNA sequence analysis. *FEMS Microbiology Reviews*, *15*(2–3), 155–173. https://doi.org/10.1111/j.1574-6976.1994.tb00132.x

Luo, G.-Z., MacQueen, A., Zheng, G., Duan, H., Dore, L. C., Lu, Z., Liu, J., Chen, K., Jia, G., Bergelson, J., & He, C. (2014). Unique features of the m6A methylome in Arabidopsis thaliana. *Nature Communications*, *5*(1), 5630. https://doi.org/10.1038/ncomms6630

Lütcke, H. A., Chow, K. C., Mickel, F. S., Moss, K. A., Kern, H. F., & Scheele, G. A. (1987). Selection of AUG initiation codons differs in plants and animals. *The EMBO Journal*, *6*(1), 43-48–48. https://doi.org/https://doi.org/10.1002/j.1460-2075.1987.tb04716.x

Lv, Y., Han, F., Liu, M., Zhang, T., Cui, G., Wang, J., Yang, Y., Yang, Y. G., & Yang, W. (2023). Characteristics of N6-methyladenosine Modification During Sexual Reproduction of Chlamydomonas reinhardtii. *Genomics, Proteomics and Bioinformatics*. https://doi.org/10.1016/j.gpb.2022.04.004

Ma, X., Ibrahim, F., Kim, E. J., Shaver, S., Becker, J., Razvi, F., Cerny, R. L., & Cerutti, H. (2020). An ortholog of the Vasa intronic gene is required for small RNA-mediated translation repression in Chlamydomonas reinhardtii. *Proceedings of the National Academy of Sciences of the United States of America*, *117*(1), 761–770. https://doi.org/10.1073/PNAS.1908356117/-/DCSUPPLEMENTAL

Ma, X., Kim, E. J., Kook, I., Ma, F., Voshall, A., Moriyama, E., & Cerutti, H. (2013). Small Interfering RNA–Mediated Translation Repression Alters Ribosome Sensitivity to Inhibition by Cycloheximide in Chlamydomonas reinhardtii. *The Plant Cell*, *25*(3), 985. https://doi.org/10.1105/TPC.113.109256

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.Journal*, *17*(1), 10–12. https://doi.org/10.14806/ej.17.1.200

Maul, J. E., Lilly, J. W., Cui, L., dePamphilis, C. W., Miller, W., Harris, E. H., & Stern, D. B. (2002). The Chlamydomonas reinhardtii plastid chromosome: islands of genes in a sea of repeats. *The Plant Cell*, *14*(11), 2659–2679. https://doi.org/10.1105/TPC.006155

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, *9*(1), 56–61.

Melnikov, S., Ben-Shem, A., Garreau De Loubresse, N., Jenner, L., Yusupova, G., & Yusupov, M. (2012). One core, two shells: Bacterial and eukaryotic ribosomes. In *Nature Structural and Molecular Biology* (Vol. 19, Issue 6, pp. 560–567). https://doi.org/10.1038/nsmb.2313

Menschaert, G., Van Criekinge, W., Notelaers, T., Koch, A., Crappé, J., Gevaert, K., & Van Damme, P. (2013). Deep Proteome Coverage Based on Ribosome Profiling Aids Mass Spectrometry-based Protein and Peptide Discovery and Provides Evidence of Alternative Translation Products and Near-cognate Translation Initiation Events. *Molecular & Cellular Proteomics*, *12*(7), 1780–1790. https://doi.org/10.1074/mcp.M113.027540

Merchant, S. S., Prochnik, S. E., Vallon, O., Harris, E. H., Karpowicz, S. J., Witman, G. B., Terry, A., Salamov, A., Fritz-Laylin, L. K., Maréchal-Drouard, L., Marshall, W. F., Qu, L.-H., Nelson, D. R., Sanderfoot, A. A., Spalding, M. H., Kapitonov, V. V, Ren, Q., Ferris, P., Lindquist, E., … Grossman, A. R. (2007). The Chlamydomonas genome reveals the evolution of key animal and plant functions. *Science (New York, N.Y.)*, *318*(5848), 245–250. https://doi.org/10.1126/science.1143609

Minagawa, J. (2011). State transitions—The molecular remodeling of photosynthetic supercomplexes that controls energy flow in the chloroplast. *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, *1807*(8), 897–905. https://doi.org/https://doi.org/10.1016/j.bbabio.2010.11.005

Minai, L., Wostrikoff, K., Wollman, F.-A., & Choquet, Y. (2006). Chloroplast Biogenesis of Photosystem II Cores Involves a Series of Assembly-Controlled Steps That Regulate Translation. *The Plant Cell*, *18*(1), 159–175. https://doi.org/10.1105/tpc.105.037705

Mishra, R. K., Datey, A., & Hussain, T. (2020). mRNA Recruiting eIF4 Factors Involved in Protein Synthesis and Its Regulation. *Biochemistry*, *59*(1), 34–46. https://doi.org/10.1021/acs.biochem.9b00788

Miyoshi, T., Nomura, T., & Uchiumi, T. (2009). Engineering and Characterization of the Ribosomal L10-L12 Stalk Complex: A Structual Element Responsible for High Turnover of the Elongation Factor G-dependent GTPase. *Journal of Biological Chemistry*, *284*(1), 85–92. https://doi.org/10.1074/jbc.M806024200

Moazed, D., & Noller, H. F. (1989). Intermediate states in the movement of transfer RNA in the ribosome. *Nature*, *342*(6246), 142–148. https://doi.org/10.1038/342142a0

Mohammad, F., Green, R., & Buskirk, A. R. (2019). A systematically-revised ribosome profiling method for bacteria reveals pauses at single-codon resolution. *ELife*, *8*, e42591. https://doi.org/10.7554/eLife.42591

Msanne, J., Vu, H. S., & Cahoon, E. B. (2021). Acyl-acyl carrier protein pool dynamics with oil accumulation in nitrogen-deprived Chlamydomonas reinhardtii microalgal cells. *Journal of the American Oil Chemists' Society*, *98*(11), 1107–1112. https://doi.org/https://doi.org/10.1002/aocs.12539

Müller, S. Y., Matthews, N. E., Valli, A. A., & Baulcombe, D. C. (2020). The small RNA locus map for Chlamydomonas reinhardtii. *PloS One*, *15*(11). https://doi.org/10.1371/JOURNAL.PONE.0242516

Mullet, J. E., & Klein, R. R. (1987). Transcription and RNA stability are important determinants of higher plant chloroplast RNA levels. *The EMBO Journal*, *6*(6), 1571-1579–1579. https://doi.org/https://doi.org/10.1002/j.1460-2075.1987.tb02402.x

Na, C. H., Barbhuiya, M. A., Kim, M.-S., Verbruggen, S., Eacker, S. M., Pletnikova, O., Troncoso, J. C., Halushka, M. K., Menschaert, G., Overall, C. M., & Pandey, A. (2018). Discovery of noncanonical translation initiation sites through mass spectrometric analysis of protein N termini. *Genome Research*, *28*(1), 25–36. https://doi.org/10.1101/gr.226050.117

Nagai, K., Oubridge, C., Kuglstatter, A., Menichelli, E., Isel, C., & Jovine, L. (2003). Structure, function and evolution of the signal recognition particle. *The EMBO Journal*, *22*(14), 3479-3485–3485. https://doi.org/https://doi.org/10.1093/emboj/cdg337

Nakagawa, S., Niimura, Y., & Gojobori, T. (2017). Comparative genomic analysis of translation initiation mechanisms for genes lacking the Shine–Dalgarno sequence in prokaryotes. *Nucleic Acids Research*, *45*(7), 3922–3931. https://doi.org/10.1093/nar/gkx124

Nellaepalli, S., Lau, A. S., & Jarvis, R. P. (2023). Chloroplast protein translocation pathways and ubiquitin-dependent regulation at a glance. *Journal of Cell Science*, *136*(18), jcs241125. https://doi.org/10.1242/jcs.241125

Neupert, J., Karcher, D., & Bock, R. (2009). Generation of Chlamydomonas strains that efficiently express nuclear transgenes. *The Plant Journal*, *57*(6), 1140–1150. https://doi.org/https://doi.org/10.1111/j.1365-313X.2008.03746.x

Nilsson, R., & van Wijk, K. J. (2002). Transient interaction of cpSRP54 with elongating nascent chains of the chloroplast-encoded D1 protein; 'cpSRP54 caught in the act'. *FEBS Letters*, *524*(1–3), 127–133. https://doi.org/https://doi.org/10.1016/S0014-5793(02)03016-8

Noller, H. F., Hoffarth, V., & Zimniak, L. (1992). Unusual Resistance of Peptidyl Transferase to Protein Extraction Procedures. *Science*, *256*(5062), 1416–1419. https://doi.org/10.1126/science.1604315

Noller, H. F., Lancaster, L., Mohan, S., & Zhou, J. (2017). Ribosome structural dynamics in translocation: yet another functional role for ribosomal RNA. *Quarterly Reviews of Biophysics*, *50*, e12. https://doi.org/DOI: 10.1017/S0033583517000117

Oh, E., Becker, A. H., Sandikci, A., Huber, D., Chaba, R., Gloge, F., Nichols, R. J., Typas, A., Gross, C. A., Kramer, G., Weissman, J. S., & Bukau, B. (2011). Selective Ribosome Profiling Reveals the Cotranslational Chaperone Action of Trigger Factor In Vivo. *Cell*, *147*(6), 1295–1308. https://doi.org/10.1016/j.cell.2011.10.044

Pan, J., Wang, Q., & Snell, W. J. (2005). Cilium-generated signaling and cilia-related disorders. *Laboratory Investigation*, *85*(4), 452–463. https://doi.org/https://doi.org/10.1038/labinvest.3700253

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel V. and Thirion, B., Grisel, O., Blondel, M., Prettenhofer P. and Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pereira, I. T., Spangenberg, L., Robert, A. W., Amorín, R., Stimamiglio, M. A., Naya, H., & Dallagiovanna, B. (2018). Polysome profiling followed by RNA-seq of cardiac differentiation stages in hESCs. *Scientific Data*, *5*(1), 180287. https://doi.org/10.1038/sdata.2018.287

Plader, W., & Sugiura, M. (2003). The Shine-Dalgarno-like sequence is a negative regulatory element for translation of tobacco chloroplast rps2 mRNA: An additional mechanism for translational control in chloroplasts. *Plant Journal*, *34*(3), 377–382. https://doi.org/10.1046/j.1365-313X.2003.01732.x

Plancke, C., Vigeolas, H., Höhner, R., Roberty, S., Emonds-Alt, B., Larosa, V., Willamme, R., Duby, F., Onga Dhali, D., Thonart, P., Hiligsmann, S., Franck, F., Eppe, G., Cardol, P., Hippler, M., & Remacle, C. (2014). Lack of isocitrate lyase in Chlamydomonas leads to changes in carbon metabolism and in the response to oxidative stress under mixotrophic growth. *The Plant Journal*, *77*(3), 404–417. https://doi.org/https://doi.org/10.1111/tpj.12392

Plöchinger, M., Schwenkert, S., von Sydow, L., Schröder, W. P., & Meurer, J. (2016). Functional Update of the Auxiliary Proteins PsbW, PsbY, HCF136, PsbN, TerC and ALB3 in Maintenance and Assembly of PSII. *Frontiers in Plant Science*, *Volume 7-2016*. https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2016.00423

Popa, A., Lebrigand, K., Paquet, A., Nottet, N., Robbe-Sermesant, K., Waldmann, R., & Barbry, P. (2016). RiboProfiling: A Bioconductor package for standard Ribo-seq pipeline processing. *F1000Research*, *5*. https://doi.org/10.12688/F1000RESEARCH.8964.1

Pospíšil, P. (2016). Production of Reactive Oxygen Species by Photosystem II as a Response to Light and Temperature Stress. *Frontiers in Plant Science*, *Volume 7-2016*. https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2016.01950

Poulhazan, A., Arnold, A. A., Mentink-Vigier, F., Muszyński, A., Azadi, P., Halim, A., Vakhrushev, S. Y., Joshi, H. J., Wang, T., Warschawski, D. E., & Marcotte, I. (2024). Molecular-level architecture of Chlamydomonas reinhardtii's glycoprotein-rich cell wall. *Nature Communications*, *15*(1), 986. https://doi.org/10.1038/s41467-024-45246-7

Prabhakar, V., Löttgert, T., Gigolashvili, T., Bell, K., Flügge, U.-I., & Häusler, R. E. (2009). Molecular and functional characterization of the plastid-localized Phosphoenolpyruvate enolase (ENO1) from Arabidopsis thaliana. *FEBS Letters*, *583*(6), 983–991. https://doi.org/https://doi.org/10.1016/j.febslet.2009.02.017

Pratt, A. J., & MacRae, I. J. (2009). The RNA-induced Silencing Complex: A Versatile Gene-silencing Machine. *Journal of Biological Chemistry*, *284*(27), 17897–17901. https://doi.org/10.1074/jbc.R900012200

Pröschold, T., Harris, E. H., & Coleman, A. W. (2005). Portrait of a Species: Chlamydomonas reinhardtii. *Genetics*, *170*(4), 1601–1610. https://doi.org/10.1534/genetics.105.044503

Rahim, M. M. A., Vigneault, F., & Zerges, W. (2016). The RNA Structure of cis-acting Translational Elements of the Chloroplast psbC mRNA in Chlamydomonas reinhardtii. *Frontiers in Plant Science*, *7*, 828. https://doi.org/10.3389/fpls.2016.00828

Raibaud, S., Vachette, P., Guillier, M., Allemand, F., Chiaruttini, C., & Dardel, F. (2003). How Bacterial Ribosomal Protein L20 Assembles with 23 S Ribosomal RNA and Its Own Messenger RNA. *Journal of Biological Chemistry*, *278*(38), 36522–36530. https://doi.org/10.1074/jbc.M304717200

Ramakrishnan, V. (2002). Ribosome Structure and the Mechanism of Translation. *Cell*, *108*(4), 557–572. https://doi.org/10.1016/S0092-8674(02)00619-0

Ramundo, S., Rahire, M., Schaad, O., & Rochaix, J.-D. (2013). Repression of Essential Chloroplast Genes Reveals New Signaling Pathways and Regulatory Feedback Loops in Chlamydomonas . *The Plant Cell*, *25*(1), 167–186. https://doi.org/10.1105/tpc.112.103051

Rochaix, J. D., Kuchka, M., Mayfield, S., Schirmer-Rahire, M., Girard-Bascou, J., & Bennoun, P. (1989). Nuclear and chloroplast mutations affect the synthesis or stability of the chloroplast psbC gene product in Chlamydomonas reinhardtii. *The EMBO Journal*, *8*(4), 1013-1021–1021. https://doi.org/https://doi.org/10.1002/j.1460-2075.1989.tb03468.x

Rodnina, M. V. (2018). Translation in prokaryotes. In *Cold Spring Harbor Perspectives in Biology* (Vol. 10, Issue 9). Cold Spring Harbor Laboratory Press. https://doi.org/10.1101/cshperspect.a032664

Rodnina, M. V, Daviter, T., Gromadski, K., & Wintermeyer, W. (2002). Structural dynamics of ribosomal RNA during decoding on the ribosome. *Biochimie*, *84*(8), 745–754. https://doi.org/https://doi.org/10.1016/S0300-9084(02)01409-8

Rodnina, M. V, Fischer, N., Maracci, C., & Stark, H. (2017). Ribosome dynamics during decoding. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *372*(1716), 20160182. https://doi.org/10.1098/rstb.2016.0182

Rodnina, M. V, & Wintermeyer, W. (2003). Peptide bond formation on the ribosome: structure and mechanism. *Current Opinion in Structural Biology*, *13*(3), 334–340. https://doi.org/https://doi.org/10.1016/S0959-440X(03)00065-4

Rodnina, M. V, & Wintermeyer, W. (2016). Protein Elongation, Co-translational Folding and Targeting. *Journal of Molecular Biology*, *428*(10, Part B), 2165–2185. https://doi.org/https://doi.org/10.1016/j.jmb.2016.03.022

Rogalski, M., Karcher, D., & Bock, R. (2008). Superwobbling facilitates translation with reduced tRNA sets. *Nature Structural & Molecular Biology*, *15*(2), 192–198. https://doi.org/10.1038/nsmb.1370

Sager, R. (1955). INHERITANCE IN THE GREEN ALGA CHLAMYDOMONAS REINHARDI. *Genetics*, *40*(4), 476–489. https://doi.org/10.1093/genetics/40.4.476

Salsi, E., Farah, E., Dann, J., & Ermolenko, D. N. (2014). Following movement of domain IV of elongation factor G during ribosomal translocation. *Proceedings of the National Academy of Sciences*, *111*(42), 15060–15065. https://doi.org/10.1073/pnas.1410873111

Samatova, E., Komar, A. A., & Rodnina, M. V. (2024). How the ribosome shapes cotranslational protein folding. *Current Opinion in Structural Biology*, *84*, 102740. https://doi.org/https://doi.org/10.1016/j.sbi.2023.102740

Sasso, S., Stibor, H., Mittag, M., & Grossman, A. R. (2018). From molecular manipulation of domesticated Chlamydomonas reinhardtii to survival in nature. *ELife*, *7*, e39233. https://doi.org/10.7554/eLife.39233

Sato, S., Nakamura, Y., Kaneko, T., Asamizu, E., & Tabata, S. (1999). Complete Structure of the Chloroplast Genome of Arabidopsis thaliana. *DNA Research*, *6*(5), 283–290. https://doi.org/10.1093/dnares/6.5.283

Scharff, L. B., Ehrnthaler, M., Janowski, M., Childs, L. H., Hasse, C., Gremmels, J., Ruf, S., Zoschke, R., & Bock, R. (2017). Shine-Dalgarno Sequences Play an Essential Role in the Translation of Plastid mRNAs in Tobacco. *The Plant Cell*, *29*(12), 3085–3101. https://doi.org/10.1105/tpc.17.00524

Schavemaker, P. E., Śmigiel, W. M., & Poolman, B. (2017). Ribosome surface properties may impose limits on the nature of the cytoplasmic proteome. *ELife*, *6*, e30084. https://doi.org/10.7554/eLife.30084

Schibich, D., Gloge, F., Pöhner, I., Björkholm, P., Wade, R. C., von Heijne, G., Bukau, B., & Kramer, G. (2016). Global profiling of SRP interaction with nascent polypeptides. *Nature*, *536*(7615), 219–223. https://doi.org/10.1038/nature19070

Schnarrenberger, C., & Martin, W. (2002). Evolution of the enzymes of the citric acid cycle and the glyoxylate cycle of higher plants. *European Journal of Biochemistry*, *269*(3), 868–883. https://doi.org/https://doi.org/10.1046/j.0014-2956.2001.02722.x

Schneider-Poetsch, T., Ju, J., Eyler, D. E., Dang, Y., Bhat, S., Merrick, W. C., Green, R., Shen, B., & Liu, J. O. (2010). Inhibition of eukaryotic translation elongation by cycloheximide and lactimidomycin. *Nature Chemical Biology*, *6*(3), 209–217. https://doi.org/10.1038/nchembio.304

Schuenemann, D., Gupta, S., Persello-Cartieaux, F., Klimyuk, V. I., Jones, J. D. G., Nussaume, L., & Hoffman, N. E. (1998). A novel signal recognition particle targets light-harvesting proteins to the thylakoid membranes. *Proceedings of the National Academy of Sciences*, *95*(17), 10312–10316. https://doi.org/10.1073/pnas.95.17.10312

Scott, W. G. (2007). Ribozymes. *Current Opinion in Structural Biology*, *17*(3), 280–286. https://doi.org/https://doi.org/10.1016/j.sbi.2007.05.003

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*.

Selbach, M., Schwanhäusser, B., Thierfelder, N., Fang, Z., Khanin, R., & Rajewsky, N. (2008). Widespread changes in protein synthesis induced by microRNAs. *Nature*, *455*(7209), 58–63. https://doi.org/10.1038/nature07228

Seth, K., Kumawat, G., Vyas, P., & Harish. (2022). The structure and functional mechanism of eyespot in Chlamydomonas. *Journal of Basic Microbiology*, *62*(10), 1169–1178. https://doi.org/https://doi.org/10.1002/jobm.202200249

Shah, P., Ding, Y., Niemczyk, M., Kudla, G., & Plotkin, J. B. (2013). Rate-Limiting Steps in Yeast Protein Translation. *Cell*, *153*(7), 1589–1601. https://doi.org/10.1016/j.cell.2013.05.049

Shajani, Z., Sykes, M. T., & Williamson, J. R. (2011). Assembly of bacterial ribosomes. *Annual Review of Biochemistry*, *80*, 501–526. https://doi.org/10.1146/annurev-biochem-062608-160432

Sharma, M. R., Dönhöfer, A., Barat, C., Marquez, V., Datta, P. P., Fucini, P., Wilson, D. N., & Agrawal, R. K. (2010). PSRP1 Is Not a Ribosomal Protein, but a Ribosome-binding Factor That Is Recycled by the Ribosome-recycling Factor (RRF) and Elongation Factor G (EF-G) 2. *Journal of Biological Chemistry*, *285*(6), 4006–4014. https://doi.org/10.1074/jbc.M109.062299

She, H., Liu, Z., Xu, Z., Zhang, H., Cheng, F., Wu, J., Wang, X., & Qian, W. (2022). Comparative chloroplast genome analyses of cultivated spinach and two wild progenitors shed light on the phylogenetic relationships and variation. *Scientific Reports*, *12*(1), 856. https://doi.org/10.1038/s41598-022-04918-4

Shi, H., & Moore, P. B. (2000). The crystal structure of yeast phenylalanine tRNA at 1.93 Å resolution: A classic structure revisited. *RNA*, *6*(8), 1091–1105. https://doi.org/DOI: 10.1017/S1355838200000364

Shine, J., & Dalgarno, L. (1974). The 3′-Terminal Sequence of Escherichia coli 16S Ribosomal RNA: Complementarity to Nonsense Triplets and Ribosome Binding Sites. *Proceedings of the National Academy of Sciences*, *71*(4), 1342–1346. https://doi.org/10.1073/pnas.71.4.1342

Simonović, M., & Steitz, T. A. (2008). Cross-crystal averaging reveals that the structure of the peptidyl-transferase center is the same in the 70S ribosome and the 50S subunit. *Proceedings of the National Academy of Sciences*, *105*(2), 500–505. https://doi.org/10.1073/pnas.0711076105

Simsek, D., Tiu, G. C., Flynn, R. A., Byeon, G. W., Leppek, K., Xu, A. F., Chang, H. Y., & Barna, M. (2017). The Mammalian Ribo-interactome Reveals Ribosome Functional Diversity and Heterogeneity. *Cell*, *169*(6), 1051-1065.e18. https://doi.org/10.1016/j.cell.2017.05.022

Smith, T., Heger, A., & Sudbery, I. (2017). UMI-tools: Modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. *Genome Research*, *27*(3), 491–499. https://doi.org/10.1101/gr.209601.116

Soler-Bistué, A., Mondotte, J. A., Bland, M. J., Val, M.-E., Saleh, M.-C., & Mazel, D. (2015). Genomic Location of the Major Ribosomal Protein Gene Locus Determines Vibrio cholerae Global Growth and Infectivity. *PLOS Genetics*, *11*(4), e1005156-. https://doi.org/10.1371/journal.pgen.1005156

Sonenberg, N., & Hinnebusch, A. G. (2007). New Modes of Translational Control in Development, Behavior, and Disease. *Molecular Cell*, *28*(5), 721–729. https://doi.org/10.1016/j.molcel.2007.11.018

Stern, D. B., Witman, George., & Harris, E. H. (2008). *The chlamydomonas sourcebook*. Academic.

Sun, Y., Bakhtiari, S., Valente-Paterno, M., Wu, Y., Nishimura, Y., Shen, W., Law, C., Dhaliwal, J., Dai, D., Bui, K. H., & Zerges, W. (2024). Chloroplast biogenesis involves spatial coordination of nuclear and organellar gene expression in Chlamydomonas. *Plant Physiology*, *196*(1), 112–123. https://doi.org/10.1093/plphys/kiae256

Sun, Y., & Jarvis, R. P. (2023). Chloroplast Proteostasis: Import, Sorting, Ubiquitination, and Proteolysis. *Annual Review of Plant Biology*, *74*(Volume 74, 2023), 259–283. https://doi.org/https://doi.org/10.1146/annurev-arplant-070122-032532

Suwannachuen, N., Leetanasaksakul, K., Roytrakul, S., Phaonakrop, N., Thaisakun, S., Roongsattham, P., Jantasuriyarat, C., Sanevas, N., & Sirikhachornkit, A. (2023). Palmelloid Formation and Cell Aggregation Are Essential Mechanisms for High Light Tolerance in a Natural Strain of Chlamydomonas reinhardtii. *International Journal of Molecular Sciences*, *24*(9). https://doi.org/10.3390/ijms24098374

Świderek, K., Marti, S., Tuñón, I., Moliner, V., & Bertran, J. (2015). Peptide Bond Formation Mechanism Catalyzed by Ribosome. *Journal of the American Chemical Society*, *137*(37), 12024–12034. https://doi.org/10.1021/jacs.5b05916

Swift, K., Chotewutmontri, P., Belcher, S., Williams-Carrier, R., & Barkan, A. (2020). Functional Analysis of PSRP1, the Chloroplast Homolog of a Cyanobacterial Ribosome Hibernation Factor. *Plants*, *9*(2). https://doi.org/10.3390/plants9020209

Tardif, M., Atteia, A., Specht, M., Cogne, G., Rolland, N., Brugière, S., Hippler, M., Ferro, M., Bruley, C., Peltier, G., Vallon, O., & Cournac, L. (2012). PredAlgo: A New Subcellular Localization Prediction Tool Dedicated to Green Algae. *Molecular Biology and Evolution*, *29*(12), 3625–3639. https://doi.org/10.1093/molbev/mss178

Trabuco, L. G., Schreiner, E., Eargle, J., Cornish, P., Ha, T., Luthey-Schulten, Z., & Schulten, K. (2010). The Role of L1 Stalk–tRNA Interaction in the Ribosome Elongation Cycle. *Journal of Molecular Biology*, *402*(4), 741–760. https://doi.org/https://doi.org/10.1016/j.jmb.2010.07.056

Traub, P., & Nomura, M. (1968). Structure and function of E. coli ribosomes. V. Reconstitution of functionally active 30S ribosomal particles from RNA and proteins. *Proceedings of the National Academy of Sciences*, *59*(3), 777–784. https://doi.org/10.1073/pnas.59.3.777

Trösch, R., Barahimipour, R., Gao, Y., Badillo-Corona, J. A., Gotsmann, V. L., Zimmer, D., Mühlhaus, T., Zoschke, R., & Willmund, F. (2018). Commonalities and differences of chloroplast translation in a green alga and land plants. *Nature Plants*, *4*(8), 564–575. https://doi.org/10.1038/s41477-018-0211-0

Turmel, M., Otis, C., & Lemieux, C. (2017). Divergent copies of the large inverted repeat in the chloroplast genomes of ulvophycean green algae. *Scientific Reports*, *7*(1), 994. https://doi.org/10.1038/s41598-017-01144-1

Uematsu, S., & Qian, S.-B. (2025). Interpreting ribosome dynamics during mRNA translation. *Journal of Biological Chemistry*, *301*(8). https://doi.org/10.1016/j.jbc.2025.110469

UniProt Consortium. (2025). UniProt: the Universal Protein Knowledgebase in 2025. *Nucleic Acids Research*, *53*(D1), D609–D617. https://doi.org/10.1093/nar/gkae1010

VanInsberghe, M., van den Berg, J., Andersson-Rolf, A., Clevers, H., & van Oudenaarden, A. (2021). Single-cell Ribo-seq reveals cell cycle-dependent translational pausing. *Nature*, *597*(7877), 561–565. https://doi.org/10.1038/s41586-021-03887-4

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, F., Dischinger, K., Westrich, L. D., Meindl, I., Egidi, F., Trösch, R., Sommer, F., Johnson, X., Schroda, M., Nickelsen, J., Willmund, F., Vallon, O., & Bohne, A.-V. (2023). One-helix protein 2 is not required for the synthesis of photosystem II subunit D1 in Chlamydomonas. *Plant Physiology*, *191*(3), 1612–1633. https://doi.org/10.1093/plphys/kiad015

Wang, Y.-H., Dai, H., Zhang, L., Wu, Y., Wang, J., Wang, C., Xu, C.-H., Hou, H., Yang, B., Zhu, Y., Zhang, X., & Zhou, J. (2023). Cryo-electron microscopy structure and translocation mechanism of the crenarchaeal ribosome. *Nucleic Acids Research*, *51*(17), 8909–8924. https://doi.org/10.1093/nar/gkad661

Watson, Z. L., Ward, F. R., Méheust, R., Ad, O., Schepartz, A., Banfield, J. F., & Cate, J. H. D. (2020). Structure of the bacterial ribosome at 2 Å resolution. *ELife*, *9*, e60482. https://doi.org/10.7554/eLife.60482

Wei, Y., & Xia, X. (2019). Unique Shine–Dalgarno Sequences in Cyanobacteria and Chloroplasts Reveal Evolutionary Differences in Their Translation Initiation. *Genome Biology and Evolution*, *11*(11), 3194–3206. https://doi.org/10.1093/gbe/evz227

Westrich, L. D., Gotsmann, V. L., Herkt, C., Ries, F., Kazek, T., Trösch, R., Armbruster, L., Mühlenbeck, J. S., Ramundo, S., Nickelsen, J., Finkemeier, I., Wirtz, M., Storchová, Z., Räschle, M., & Willmund, F. (2021). The versatile interactome of chloroplast ribosomes revealed by affinity purification mass spectrometry. *Nucleic Acids Research*, *49*(1), 400–415. https://doi.org/10.1093/nar/gkaa1192

Wilson, D. N., & Doudna Cate, J. H. (2012). The Structure and Function of the Eukaryotic Ribosome. *Cold Spring Harbor Perspectives in Biology*, *4*(5), a011536. https://doi.org/10.1101/cshperspect.a011536

Wostrikoff, K., Girard-Bascou, J., Wollman, F., & Choquet, Y. (2004). Biogenesis of PSI involves a cascade of translational autoregulation in the chloroplast of Chlamydomonas. *The EMBO Journal*, *23*(13), 2696-2705–2705. https://doi.org/https://doi.org/10.1038/sj.emboj.7600266

Xu, D., & Wang, Y. (2021). Protein-free ribosomal RNA scaffolds can assemble poly-lysine oligos from charged tRNA fragments. *Biochemical and Biophysical Research Communications*, *544*, 81–85. https://doi.org/https://doi.org/10.1016/j.bbrc.2021.01.036

Yamaguchi, K., & Subramanian, A. R. (2000). The Plastid Ribosomal Proteins: IDENTIFICATION OF ALL THE PROTEINS IN THE 50 S SUBUNIT OF AN ORGANELLE RIBOSOME (CHLOROPLAST) *. *Journal of Biological Chemistry*, *275*(37), 28466–28482. https://doi.org/10.1074/jbc.M005012200

Yamaguchi, K., & Subramanian, A. R. (2003). Proteomic identification of all plastid-specific ribosomal proteins in higher plant chloroplast 30S ribosomal subunit. *European Journal of Biochemistry*, *270*(2), 190–205. https://doi.org/https://doi.org/10.1046/j.1432-1033.2003.03359.x

Yamaguchi, K., von Knoblauch, K., & Subramanian, A. R. (2000). The Plastid Ribosomal Proteins: IDENTIFICATION OF ALL THE PROTEINS IN THE 30 S SUBUNIT OF AN ORGANELLE RIBOSOME (CHLOROPLAST) *. *Journal of Biological Chemistry*, *275*(37), 28455–28465. https://doi.org/10.1074/jbc.M004350200

Yang, D., Tewary, P., de la Rosa, G., Wei, F., & Oppenheim, J. J. (2010). The alarmin functions of high-mobility group proteins. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms*, *1799*(1), 157–163. https://doi.org/https://doi.org/10.1016/j.bbagrm.2009.11.002

Yuan, J., Kight, A., Goforth, R. L., Moore, M., Peterson, E. C., Sakon, J., & Henry, R. (2002). ATP Stimulates Signal Recognition Particle (SRP)/FtsY-supported Protein Integration in Chloroplasts *. *Journal of Biological Chemistry*, *277*(35), 32400–32404. https://doi.org/10.1074/jbc.M206192200

Yusupova, G., Jenner, L., Rees, B., Moras, D., & Yusupov, M. (2006). Structural basis for messenger RNA movement on the ribosome. *Nature*, *444*(7117), 391–394. https://doi.org/10.1038/nature05281

Yusupova, G. Z., Yusupov, M. M., Cate, J. H. D., & Noller, H. F. (2001). The path of messenger RNA through the ribosome. *Cell*, *106*(2), 233–241. https://doi.org/10.1016/S0092-8674(01)00435-4

Zaug, A. J., & Cech, T. R. (1986). The Intervening Sequence RNA of Tetrahymena Is an Enzyme. *Science*, *231*(4737), 470–475. https://doi.org/10.1126/science.3941911

Zerges, W., Auchincloss, A. H., & Rochaix, J.-D. (2003). Multiple Translational Control Sequences in the 5′ Leader of the Chloroplast psbC mRNA Interact With Nuclear Gene Products in Chlamydomonas reinhardtii. *Genetics*, *163*(3), 895–904. https://doi.org/10.1093/genetics/163.3.895

Zerges, W., Girard-Bascou, J., & Rochaix, J. D. (1997). Translation of the chloroplast psbC mRNA is controlled by interactions between its 5' leader and the nuclear loci TBC1 and TBC3 in Chlamydomonas reinhardtii. *Molecular and Cellular Biology*, *17*(6), 3440–3448. http://www.ncbi.nlm.nih.gov/pubmed/9154843

Zhang, L., Paakkarinen, V., van Wijk, K. J., & Aro, E.-M. (1999). Co-translational Assembly of the D1 Protein into Photosystem II. *Journal of Biological Chemistry*, *274*(23), 16062–16067. https://doi.org/10.1074/jbc.274.23.16062

Zhang, Y., Li, H., Shen, Y., Wang, S., Tian, L., Yin, H., Shi, J., Xing, A., Zhang, J., Ali, U., Sami, A., Chen, X., Gao, C., Zhao, Y., Lyu, Y., Wang, X., Chen, Y., Tian, Z., Wu, S.-B., & Wu, L. (2024). Readthrough events in plants reveal plasticity of stop codons. *Cell Reports*, *43*(2), 113723. https://doi.org/https://doi.org/10.1016/j.celrep.2024.113723

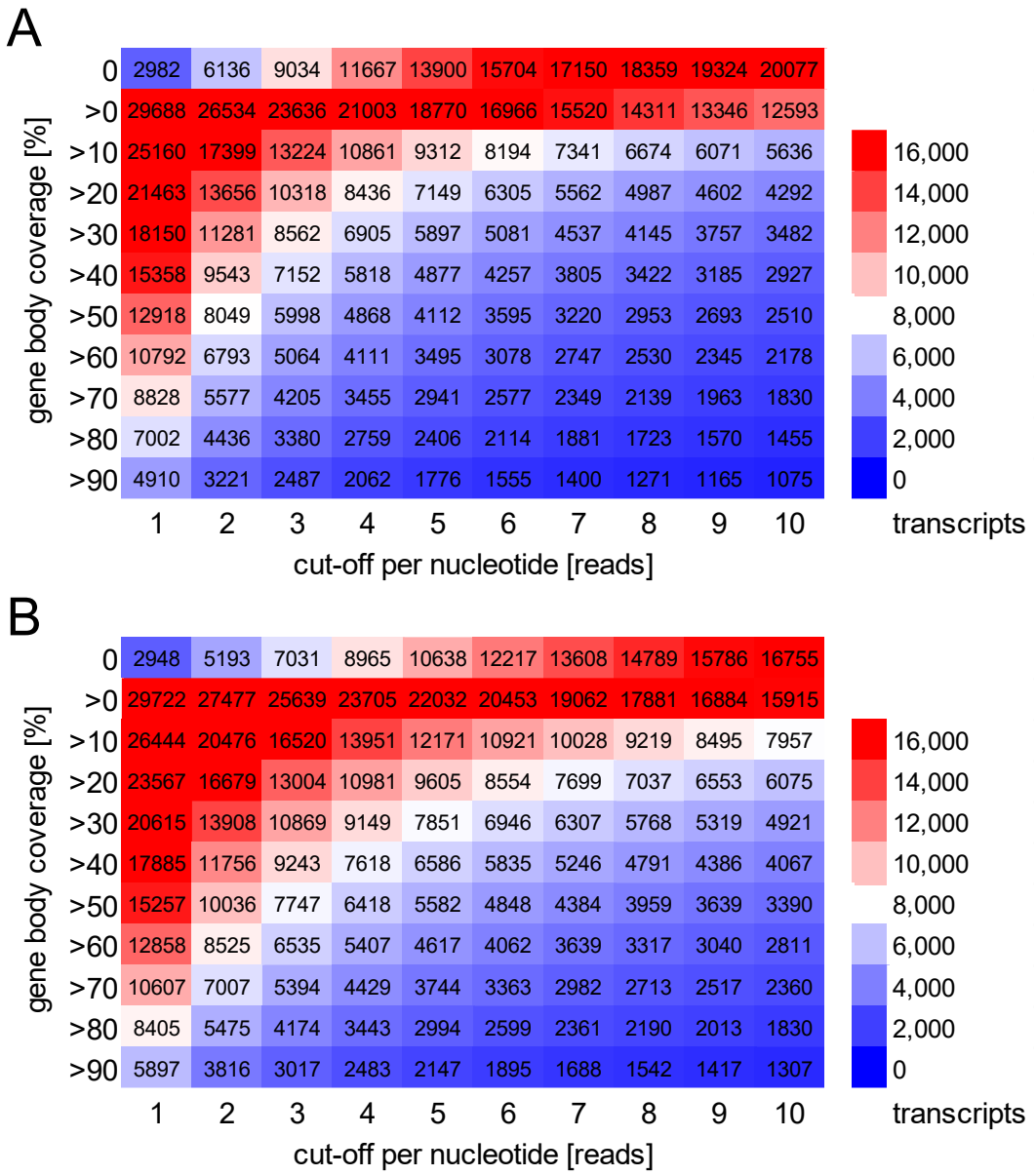
Ziehe, D., Dünschede, B., & Schünemann, D. (2018). Molecular mechanism of SRP-dependent light-harvesting protein transport to the thylakoid membrane in plants. *Photosynthesis Research*, *138*(3), 303–313. https://doi.org/10.1007/s11120-018-0544-6

Zimorski, V., Ku, C., Martin, W. F., & Gould, S. B. (2014). Endosymbiotic theory for organelle origins. *Current Opinion in Microbiology*, *22*, 38–48. https://doi.org/https://doi.org/10.1016/j.mib.2014.09.008

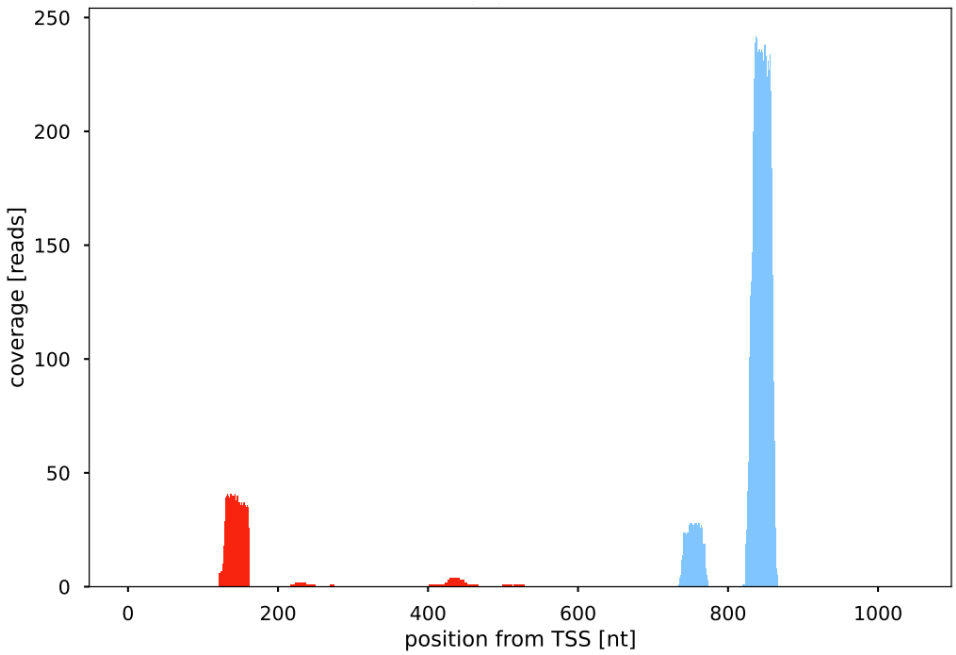
Zoschke, R., & Bock, R. (2018). Chloroplast Translation: Structural and Functional Organization, Operational Control, and Regulation. *Plant Cell Advance Publication. Published On*. https://doi.org/10.1105/tpc.18.00016

8 Appendix

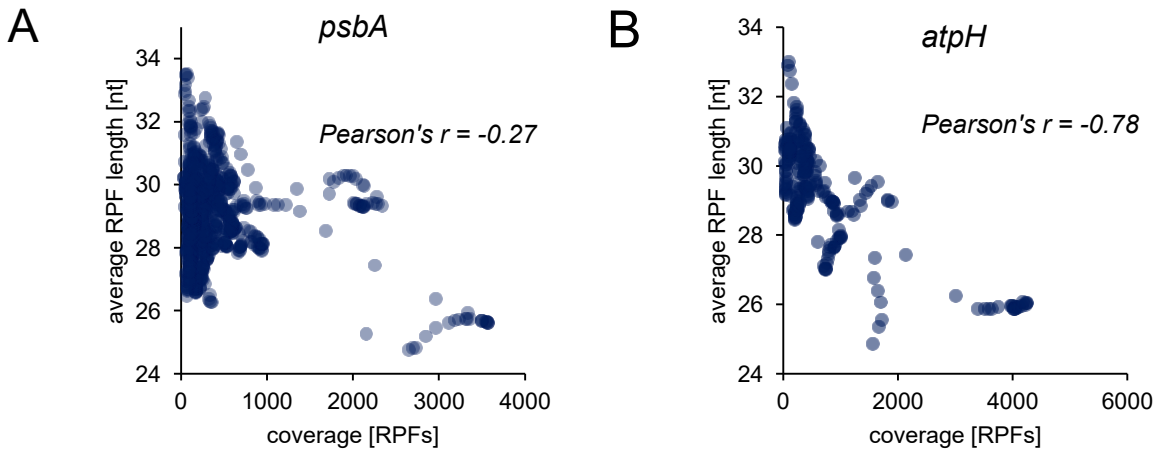
8.1 Supplementary Figures



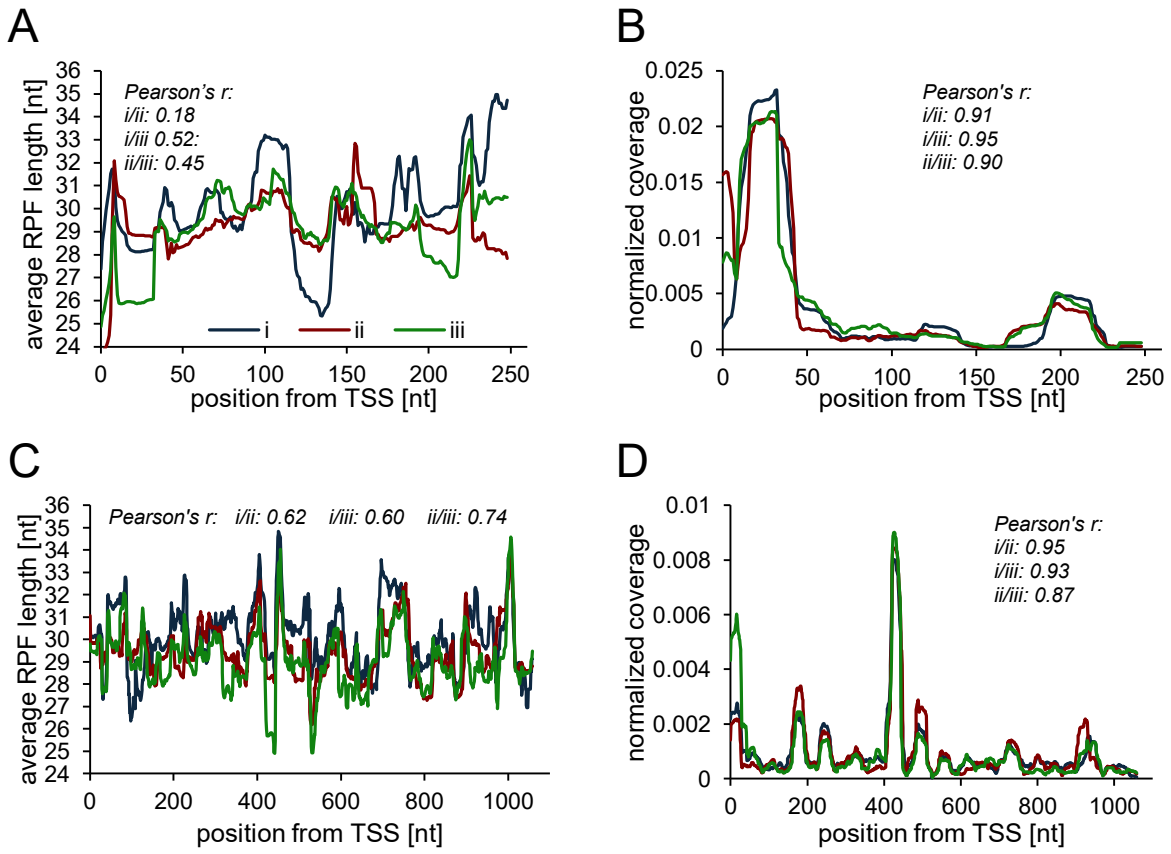
Supplementary Figure 1: Cumulative heat maps visualizing the number of transcripts covered to a certain minimal extent (rows) by a certain minimal number of reads (columns) in sample i (A) and sample ii (B). Supporting Figure 3-4.



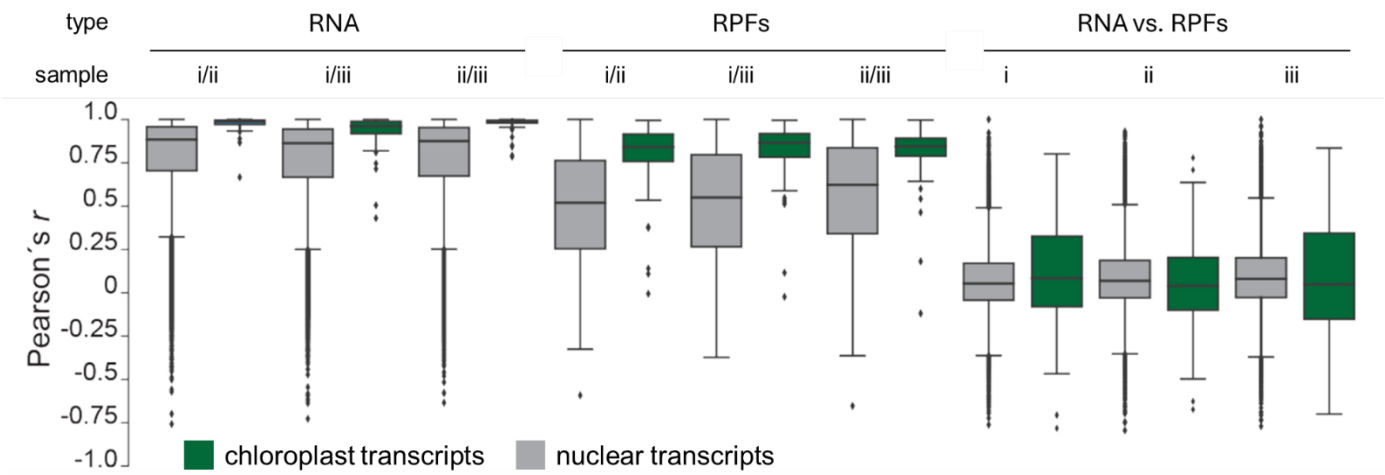
Supplementary Figure 2: Ribosome profile of chloroplast transcript *rpl20* from sample iii.



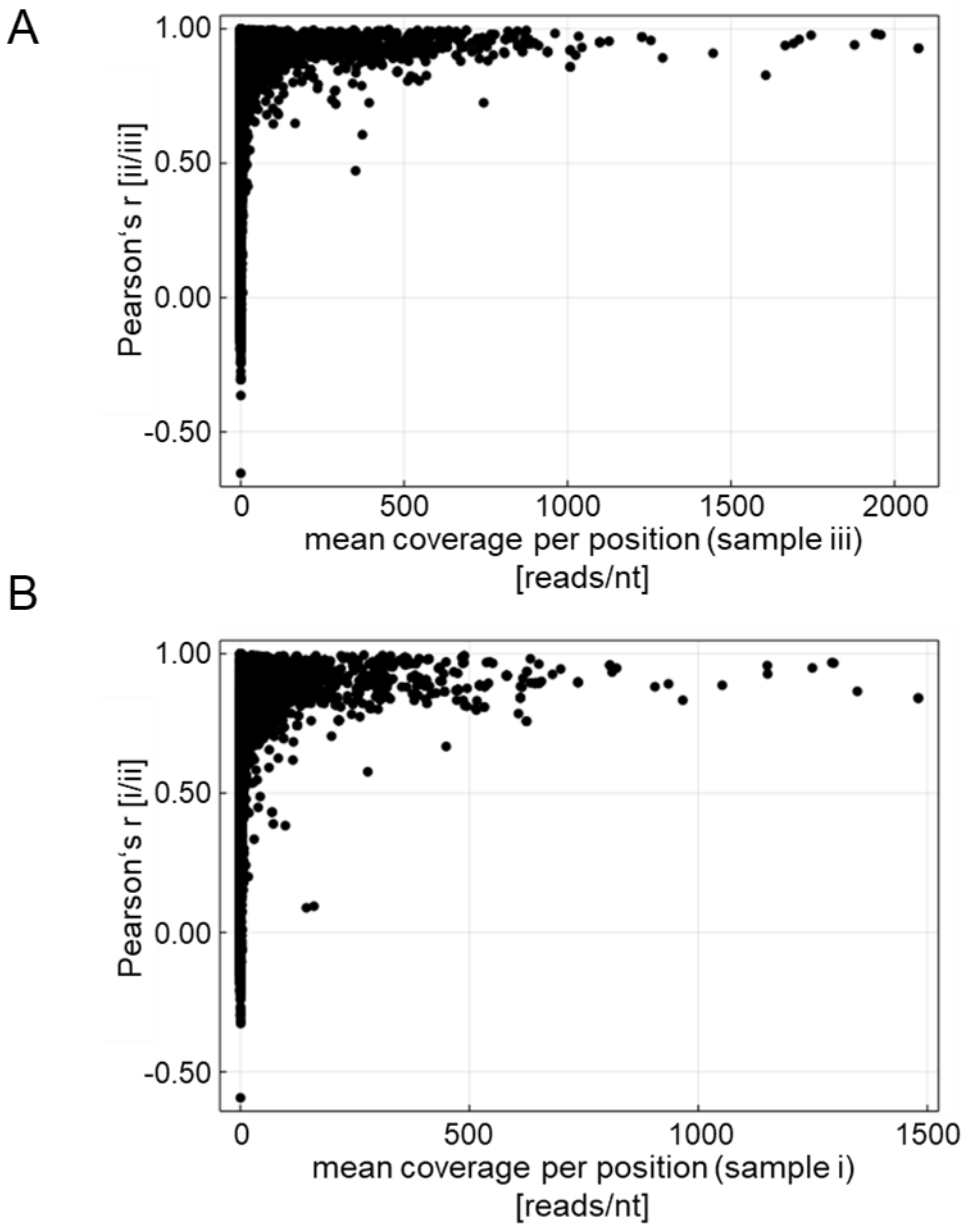
Supplementary Figure 3: Scatter plots of average RPF length and coverage of chloroplast transcripts *psbA* (A) and *atpH* (B) in sample iii. Average RPF length was calculated position-wise as the mean of lengths of all RPFs mapping to respective positions. Coverage was calculated position-wise as the total number of RPFs that were mapped to respective positions.



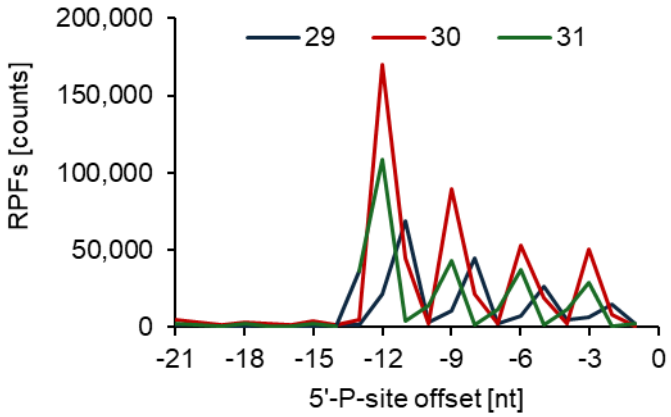
Supplementary Figure 4: Comparison of average RPF length distribution along transcripts and ribosome profiles between samples i – iii for chloroplast transcripts *psbA* and *atpH*. (A) Distribution of average RPF length along *atpH*. (B) Normalized ribosome profiles of *atpH*. (C) Distribution of average RPF length along *psbA*. (D) Normalized ribosome profiles of *psbA*. Colors are indicated in the legend of (A) and apply to all plots. Average RPF length was calculated position-wise as the mean of lengths of all RPFs mapping to respective position. Ribosome profiles were normalized by the sum of RPF counts of all positions. TSS: transcription start site.



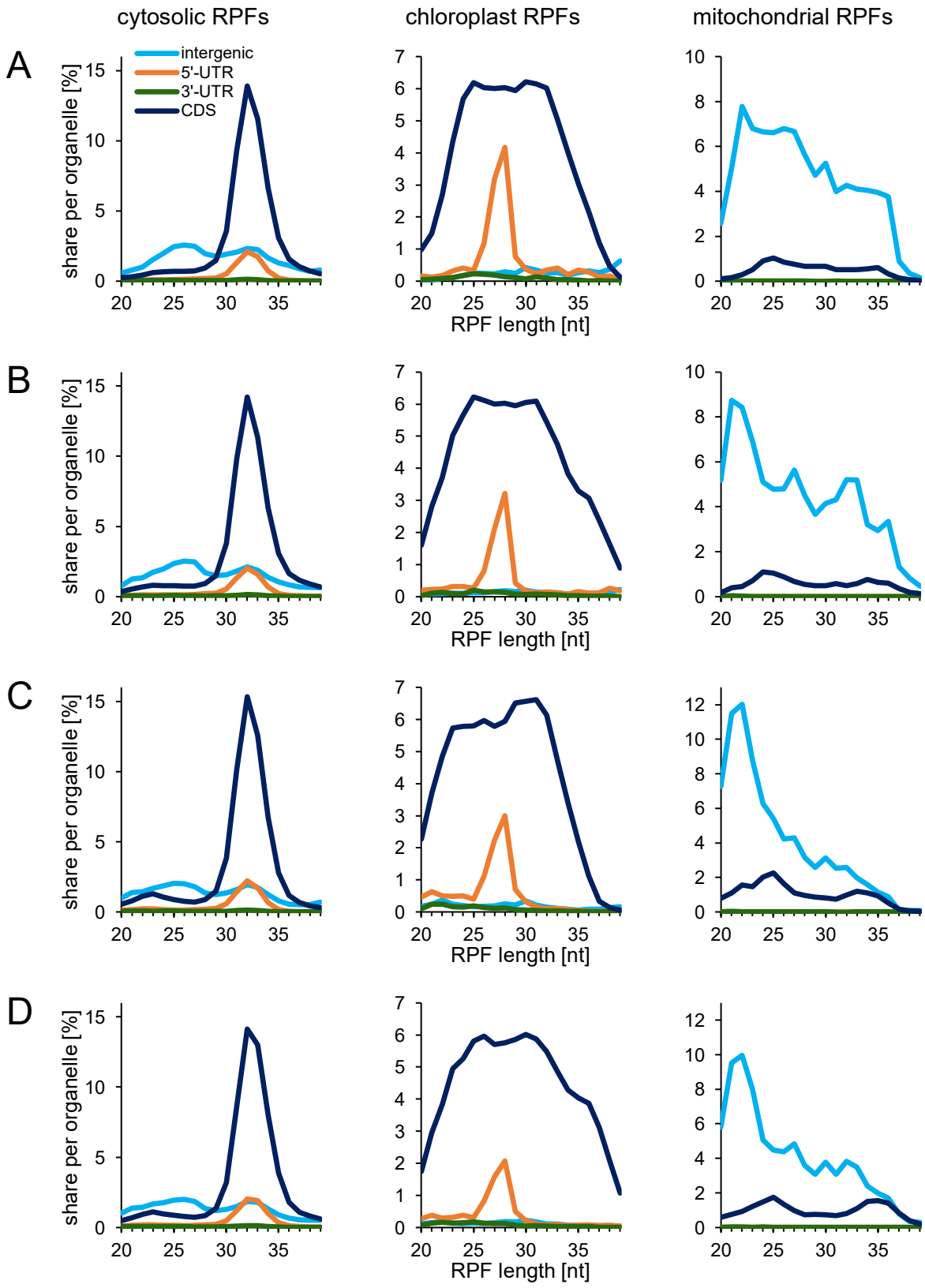
Supplementary Figure 5: Box plots of Pearson's correlation coefficients of ribosome profiles and RNA-Seq gene body coverage between samples and across each other. Correlations were split between nuclear and chloroplast transcripts.



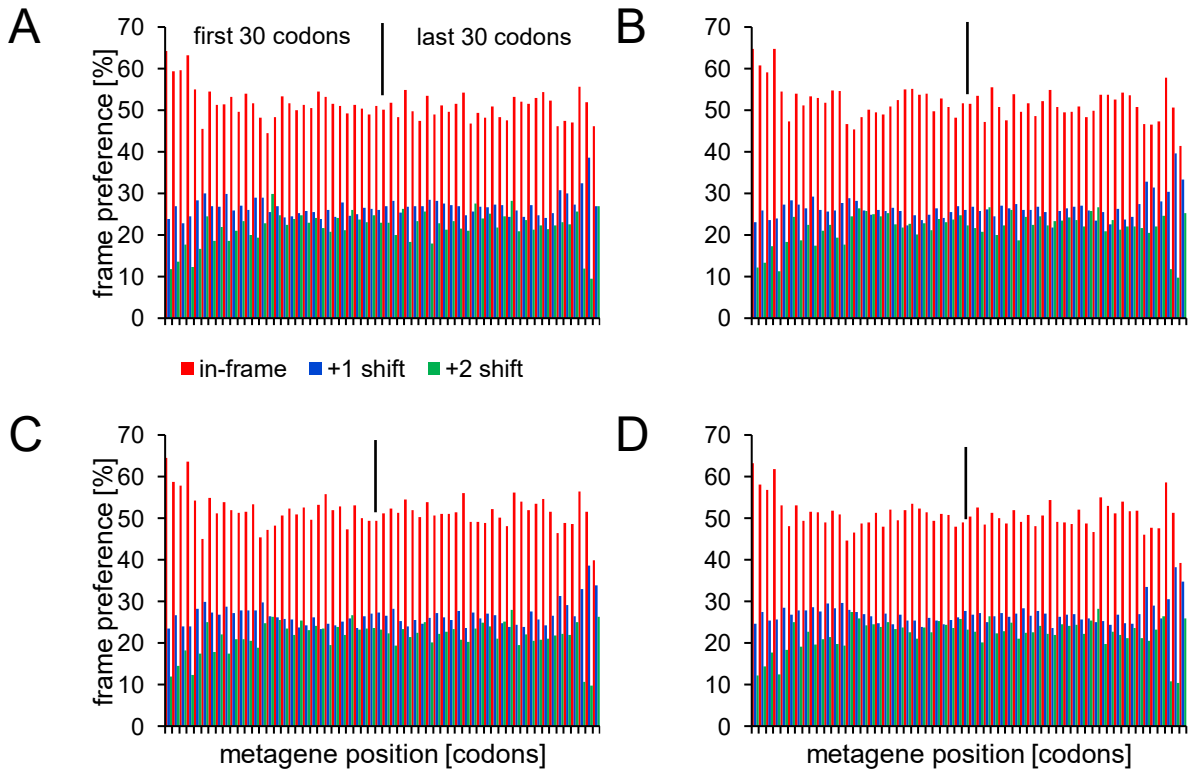
Supplementary Figure 6: Relationship between coverage and correlation of ribosome profiles between samples. Average coverage per position was calculated as the total number of RPFs mapping to an ORF divided by its total length. (A) Transcript-wise Pearson's correlation coefficients of ribosome profiles between sample ii and iii. (B) Transcript-wise Pearson's correlation coefficients of ribosome profiles between sample i and ii.



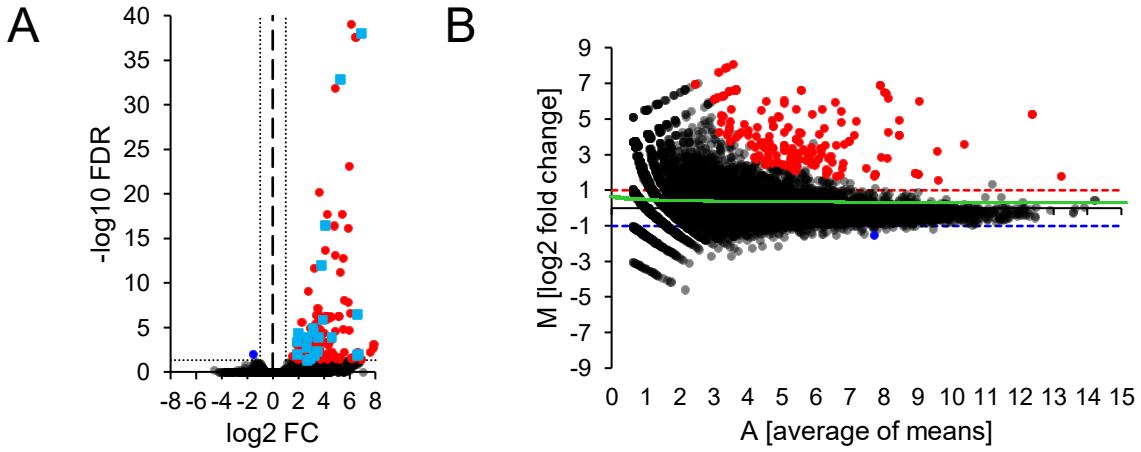
Supplementary Figure 7: Detailed metagene view on triplet-wise movement of cytosolic ribosomes. Red, blue and green lines represent accumulations of RPFs at given metagene transcript coordinates for the cytosolic RPF species of 29, 30 and 31 nt length in sample iii, respectively.



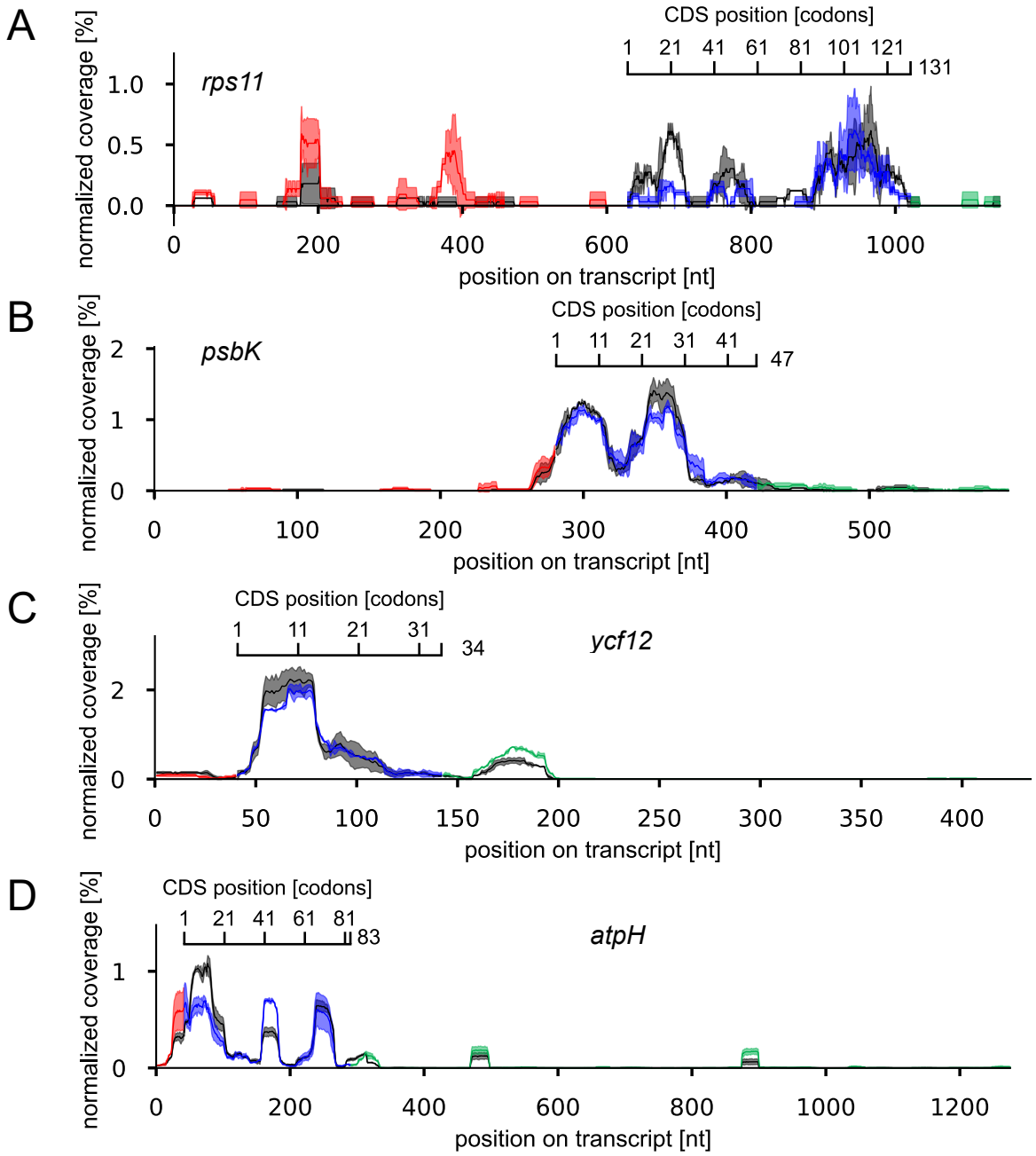
Supplementary Figure 8: RPF length distributions of individual samples in uS11c knock-down experiment. (A) non-induced replicate A, (B) non-induced replicate B, (C) induced replicate A, (D) induced replicate B. RPF proportions were split according to their organellar origin (columns) and to their biotypes (colors indicated in panel A).



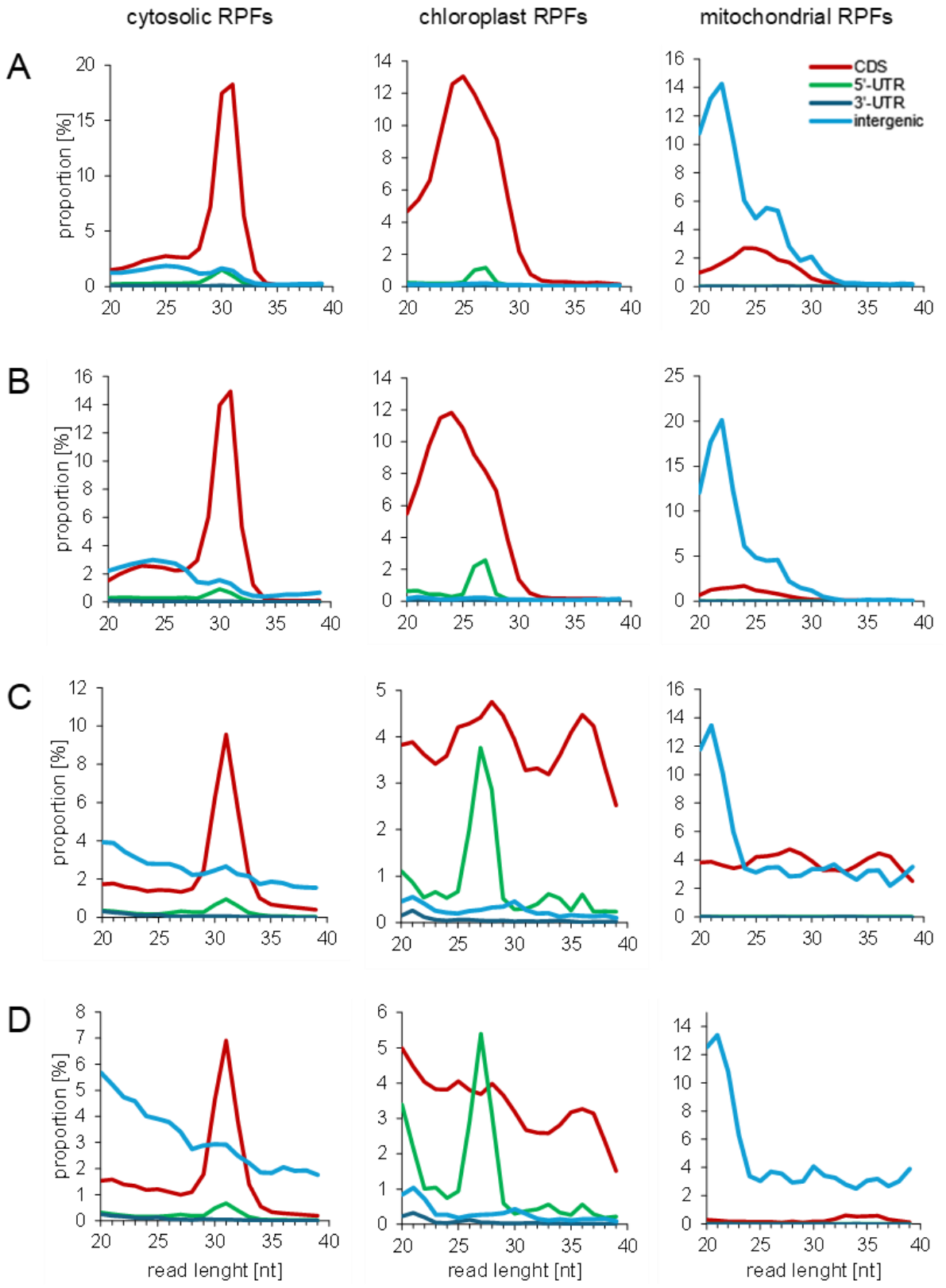
Supplementary Figure 9: Metagene frame-preference of cytosolic RPFs in individual replicates of the uS11c knock-down experiment. (A) non-induced replicate A, (B) non-induced replicate B, (C) induced replicate A, (D) induced replicate B Black lines separate the first 30 metagene codons from the last 30 metagene codons. Colors are indicated in A. Frame preference was calculated codon-wise as the proportion of RPFs mapping to the respective metagene codon in respective frame. Frame preference was calculated considering the 2000 top-translated transcripts.



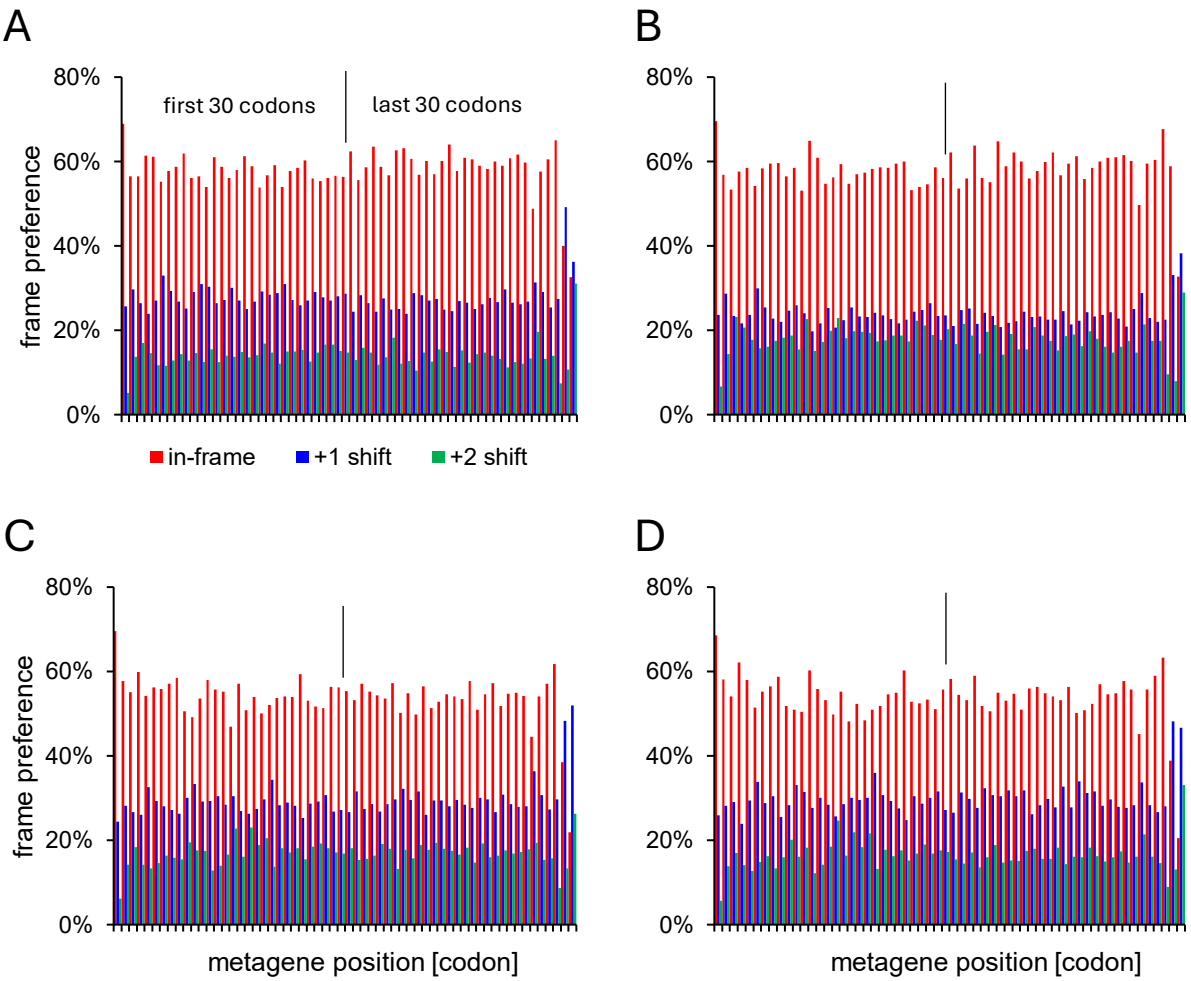
Supplementary Figure 10: DE-analysis of cpSRP54-selective ribosome profiling experiment. (A) Volcano plot representing transcripts in seRP experiment by Benjamini-Hochberg corrected p-values (-log10 FDR) versus log2 fold change (log2 FC) of ctrl / induced samples. Black dots represent transcripts not differentially expressed on the translational level, blue dots represent depleted transcripts, red dots represent enriched transcripts with annotated non-chloroplast localization. Cyan dots represent transcripts enriched with chloroplast or unknown localization. Dashed vertical line indicates 0 change, dotted grey lines represent borders of significance on both axes. (B) MA-plot representing transcripts in seRP experiment by their log2 average expression between both conditions and their log2 fold change. Colors like A. Blue and red dotted lines represent the borders of significance on the log2-fold change axis. Green line represents the LOWESS curve calculated on the data.



Supplementary Figure 11: (A – D) Comparative full length ribosome profiles of chloroplast transcripts negatively affected by uS11c depletion. Transcript names are indicated in italic letters. Black lines represent the average reference profile of non-induced cultures. Colored lines represent the average profiles of induced cultures. Red parts represent 5'-UTRs, blue parts represent coding sequences and green parts represent 3'-UTRs. Shaded areas represent the standard deviation-based error band. Scale bars at the top indicate the codon-wise position along the coding sequence (CDS).



Supplementary Figure 12: Relative RPF length distribution in samples of cpSRp54-selective ribosome profiling experiment separately for chloroplastic, nuclear and mitochondrial RPFs. (A) Control sample A, (B) control samble B, (C) cpSRP54-pulldown A, (D) cpSRP54-pulldown B. Colors indicate different biotypes of RPFs as presented in panel A.Proportions were calculated as the number of RPFs divided by the total number of RPFs for the respective subcellular compartment.



Supplementary Figure 13: Metagene frame-preference of cytosolic RPFs in individual replicates of the cpSRP54-selective ribosome profiling experiment. (A) Control sample A, (B) Control sample B, (C) cpSRP54-pulldown A, (D) cpSRP54 pulldown B Black lines separate the first 30 metagene codons from the last 30 metagene codons. Colors are indicated in A. Frame preference was calculated codon-wise as the proportion of RPFs mapping to the respective metagene codon in respective frame. Frame preference was calculated considering the 2000 top-translated transcripts.

8.2 Supplementary Tables

Supplementary Table 1: Transcripts enriched in cpSRP54-selective ribosome profiling encoding proteins predicted to be non-chloroplast localized. Transcripts with failed PredAlgo prediction are shaded in grey. Transcripts are sorted according to enrichment strength in decreasing order. FDR represents the Benjamini-Hochberg corrected p-values.

#	transcript name	log2-fold enrichment	FDR
1	**Cre11.g478400**	7.74	7.01E-04
2	**Cre10.g801109**	7.56	8.88E-04
3	**CMT1B**	7.51	1.99E-03
4	**Cre07.g800805**	7.30	5.66E-03
5	**Cre08.g358543**	6.57	3.44E-02
6	**Cre07.g325760**	6.57	4.02E-02
7	**Cre16.g654000**	6.41	2.29E-38
8	**Cre12.g553300**	6.34	2.45E-38
9	**Cre17.g705778**	6.30	5.66E-03
10	**Cre12.g801471**	6.05	8.36E-40
11	**Cre17.g701600**	5.98	2.82E-02
12	**Cre12.g519600**	5.93	2.00E-02
13	**Cre05.g238550**	5.93	7.52E-24
14	**Cre07.g329500**	5.92	2.76E-02
15	**Cre12.g491550**	5.84	2.79E-02
16	**Cre04.g800499**	5.80	2.64E-07
17	**Cre06.g800695**	5.76	3.23E-02
18	**Cre07.g325744**	5.76	6.48E-17
19	**Cre02.g079926**	5.76	3.30E-02
20	**Cre03.g800394**	5.70	2.04E-05
21	**Cre02.g097300**	5.62	1.46E-08
22	**Cre12.g540900**	5.34	7.85E-09
23	**Cre06.g296950**	5.33	1.72E-13
24	**SPP1C**	5.25	2.06E-18
25	**Cre16.g678661**	5.23	5.66E-03
26	**Cre05.g800522**	5.21	1.52E-05
27	**Cre03.g190250**	5.07	7.59E-12
28	**Cre10.g464037**	4.92	5.79E-07
29	**Cre12.g496850**	4.90	2.35E-02
30	**Cre13.g588271**	4.82	3.71E-02
31	**Cre11.g467653**	4.82	4.95E-07
32	**Cre06.g295600**	4.81	1.41E-32
33	**Cre15.g801691**	4.73	7.46E-14
34	**Cre01.g064727**	4.64	3.64E-17
35	**Cre06.g800755**	4.63	2.79E-05
36	**SCPL49**	4.52	1.91E-02
37	**MMP24**	4.49	2.92E-02
38	**Cre15.g801843**	4.39	5.90E-03
39	**Cre03.g155700**	4.31	7.13E-07
40	**Cre36.g759597**	4.30	5.28E-07
41	**MTP4**	4.23	3.84E-04
42	**Cre10.g801097**	4.21	1.69E-02
43	**TTL12**	4.14	2.12E-18
44	**Cre12.g557300**	4.05	9.83E-04
45	**Cre14.g801619**	4.03	2.30E-02
46	**Cre07.g347450**	4.02	4.18E-04
47	**Cre07.g335000**	4.02	1.27E-06
48	**MARS1**	3.99	2.73E-17
49	**Cre13.g591073**	3.97	8.41E-03
50	**Cre12.g515400**	3.96	2.28E-14
51	**Cre01.g012850**	3.94	5.47E-07
52	**ARS73A**	3.92	5.17E-07
53	**Cre19.g751297**	3.90	4.32E-02
54	**Cre10.g801128**	3.85	2.17E-02
55	**Cre16.g801919**	3.82	1.14E-02
56	**Cre16.g682026**	3.80	1.14E-02
57	**Cre08.g360500**	3.77	2.35E-02
58	**Cre09.g393617**	3.74	5.27E-04
59	**Cre35.g802212**	3.72	4.37E-05
60	**Cre03.g205249**	3.67	1.79E-02
61	**PHC35**	3.59	6.43E-21
62	**Cre12.g525400**	3.43	6.78E-04
63	**Cre09.g399812**	3.43	1.76E-05
64	**Cre12.g546450**	3.43	4.15E-04
65	**GSL3**	3.43	4.59E-03
66	**Cre07.g325749**	3.40	3.81E-02
67	**Cre09.g402108**	3.39	2.52E-07
68	**Cre10.g457331**	3.38	1.22E-02
69	**TRP6**	3.36	2.15E-06
70	**Cre06.g254650**	3.32	7.84E-08
71	**Cre13.g590200**	3.30	2.11E-03
72	**Cre02.g104250**	3.30	8.15E-03
73	**Cre16.g686286**	3.23	3.85E-05
74	**Cre12.g537671**	3.17	1.97E-12
75	**Cre10.g441850**	3.16	3.77E-07
76	**Cre11.g467608**	3.15	6.35E-03
77	**Cre16.g673600**	3.14	5.82E-04
78	**POC5**	3.10	3.33E-03
79	**Cre10.g457400**	3.09	4.39E-06
80	**Cre16.g684379**	3.09	7.53E-03
81	**Cre01.g020182**	3.07	7.01E-04
82	**Cre10.g435750**	3.07	6.35E-03
83	**FAP21**	2.94	9.46E-06
84	**Cre02.g105250**	2.86	4.65E-04
85	**SAK1**	2.82	5.66E-03
86	**Cre09.g412175**	2.79	1.31E-02
87	**Cre10.g451752**	2.76	3.23E-04
88	**Cre07.g333000**	2.76	7.51E-05
89	**Cre12.g499450**	2.75	1.25E-05
90	**Cre02.g144001**	2.73	5.50E-04
91	**Cre04.g219150**	2.71	8.28E-10
92	**Cre02.g146300**	2.70	6.98E-03
93	**Cre15.g801769**	2.69	5.27E-04
94	**Cre01.g026350**	2.68	1.13E-02
95	**Cre13.g591300**	2.58	2.72E-02
96	**Cre04.g226700**	2.58	7.01E-04
97	**Cre14.g632750**	2.54	7.84E-04
98	**Cre14.g611750**	2.54	1.98E-02
99	**Cre13.g801566**	2.53	5.11E-04
100	**Cre17.g741750**	2.52	5.54E-04
101	**MRPS26**	2.46	4.78E-03
102	**Cre13.g590850**	2.44	3.14E-02
103	**FXL7**	2.40	2.97E-02
104	**Cre13.g589150**	2.36	4.09E-02
105	**Cre08.g382250**	2.33	2.32E-03
106	**PHC38**	2.33	4.43E-03
107	**Cre13.g590150**	2.33	2.40E-02
108	**Cre03.g800364**	2.29	5.81E-03
109	**Cre12.g545500**	2.28	3.23E-04
110	**Cre03.g198750**	2.24	3.44E-02
111	**RAD9**	2.21	3.81E-02
112	**CGL153**	2.21	9.29E-03
113	**Cre01.g051550**	2.20	2.37E-04
114	**Cre17.g707137**	2.19	4.56E-02
115	**CNK3**	2.14	7.26E-03
116	**RSEP3**	2.12	3.30E-03
117	**Cre03.g800279**	2.12	2.70E-06
118	**Cre16.g661950**	2.05	1.76E-02
119	**Cre01.g051300**	2.04	4.53E-04
120	**Cre16.g657700**	1.97	2.40E-02
121	**FAP102**	1.94	6.56E-04
122	**OPR9**	1.90	2.43E-02
123	**Cre01.g046237**	1.87	2.83E-02
124	**Cre02.g142166**	1.86	2.40E-03
125	**FAP327**	1.70	6.37E-03
126	**CTP4**	1.69	4.26E-02
127	**Cre17.g705500**	1.66	8.94E-04
129	**Cre14.g630750**	1.62	1.63E-02
129	**FAP233**	1.53	1.79E-02

Supplementary Table 2: Counts and percentage proportion of chloroplast, cytosolic and mitochondrial RPFs of all Ribo-Seq samples in this study per sample, respectively.

sample	chloroplast RPFs		cytosolic RPFs		mitochondrial RPFs		total
	count	percentage	count	percentage	count	percentage	
sample i	203576	3.37%	5824825	96.34%	17645	0.29%	6046046
sample ii	232866	2.58%	8769083	97.10%	29080	0.32%	9031029
sample iii	432460	4.30%	9585464	95.27%	43810	0.44%	10061734
S11kd2 ctrl A	44325	1.48%	2928316	97.75%	22979	0.77%	2995620
S11kd5 ctrl B	46629	1.86%	2442525	97.32%	20717	0.83%	2509871
S11kd2 induced A	76329	1.57%	4751568	97.88%	26758	0.55%	4854655
S11kd5 induced B	105872	1.95%	5300152	97.41%	35086	0.64%	5441110
cpSRP54 ctrl A	219831	4.81%	4327482	94.67%	23894	0.52%	4571207
cpSRP54 ctrl B	98207	3.13%	3018111	96.21%	20632	0.66%	3136950
cpSRP54 pulldown A	60079	4.32%	1310349	94.15%	21364	1.53%	1391792
cpSRP54 pulldown B	29986	1.92%	1517293	96.95%	17794	1.14%	1565073

8.3 Ribo-Seq functions source code

```
import pysam as ps
import pandas as pd
import math
import pickle
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import multiprocessing as mp
import plotly.express as px
import itertools as it
from time import time

'''
The transcript class is the heart of the module. It can be instantiated for an arbitrary
genomic location but is optimized to be instantiated on a whole transcriptome prepared by
the "prepare_genomic_coordinates" function line by line.

IMPORTANT: COMMENTS ABOUT FEATURES HERE MEAN GENOMIC FEATURES, WHICH ARE 5'-UTR, CDS (coding
sequence) AND 3'-UTR!!!
'''
class Transcript:
    def __init__(self, chromosome, strand, five_utr, three_utr, codings, exons, name,
identifier, samfile, info, sort=True):
        '''ALL COORDINATES HAVE TO BE ORGANIZED IN TUPLES! The coordinates for exons of 5'-
UTR (five_utr), 3'-UTR (three_utr) and CDS (codings) have to be given
        as list of tuples with the coordinates of the respective exon like this:
[(start1,stop1),(start2,stop2),(start3,stop3)]'''
        self.chromosome = chromosome #has to be a chromosomal identifier that occurs in the
BAM file to parse
        if self.chromosome == "mating_type_plus":
            self.chromosome = "chromosome_06"
        self.strand = strand #as to be either + or -
        self.five_utr = five_utr
        self.three_utr = three_utr
        self.codings = codings
        self.exons = exons
        self.name = name #a trivial name of the transcript
        self.identifier = identifier #an unmistakable unique identifier for the transcript
        self.sort = sort #True or False
        self.samfile = samfile #Has to be a pysam.AlignmentFile object
        self.info = info #contains the whole description of this transcript from the
annotation file

        #check coordinates for correct order. Has to be (start, stop) in all cases
        #check coordinates for 5'UTR
        if self.sort == True:
            newfive_utr = []

            for i in self.five_utr:
                if i[0] > i[1]:
                    newfive_utr.append((i[1]-1, i[0]))
                else:
                    newfive_utr.append(i)

            if self.strand == "+":
                self.five_utr = sorted(newfive_utr)
            elif self.strand == "-":
                self.five_utr = sorted(newfive_utr, reverse=True)

            #sort the order of CDS exons according to + (like (1,10),(12,30)...) or
            #-(like (1500,2000),(1000,1200)...) strandedness

            newcodings = []

            for i in self.codings:
                if i[0] > i[1]:
                    newcodings.append((i[1], i[0]))
                else:
                    newcodings.append(i)

            if self.strand == "+":
                self.codings = sorted(newcodings)
            elif self.strand == "-":
                self.codings = sorted(newcodings, reverse=True)
```

```
            #check coordinates for 3'UTR

            newthree_utr = []

            for i in self.three_utr:
                if i[0] > i[1]:
                    newthree_utr.append((i[1], i[0]))
                else:
                    newthree_utr.append(i)

            if self.strand == "+":
                self.three_utr = sorted(newthree_utr)
            elif self.strand == "-":
                self.three_utr = sorted(newthree_utr, reverse=True)

            #check coordinates for exons

            newexons = []

            for i in self.exons:
                if i[0] > i[1]:
                    newexons.append((i[1], i[0]))
                else:
                    newexons.append(i)

            if self.strand == "+":
                self.exons = sorted(newexons)
            elif self.strand == "-":
                self.exons = sorted(newexons, reverse=True)

        #save the total number of reads in the BAM to totreads and divided by 1 M to mreads
for later normalizations
        self.totreads = self.samfile.mapped
        self.mreads = self.totreads/1000000

    '''This method queries the BAM file for reads mapping to the coordinates given to the
Transcript instance with regard to CDS and UTRs. All metrics calculated here therefore
    rather resemble the way you would go in a normal RNA-Seq dataset.'''
    def fulltranscript(self, get_readnames=False, calc_gbc=False):
        samfile = self.samfile #The BAM or SAM file to query
        if len(self.codings) == 0:
            coordinates = self.exons
        else:
            coordinates = [self.five_utr, self.codings, self.three_utr] #all genomic parts
are lists of tuples representing the exons of the part
        coverage = {} #The coverage of each position of the transcript will be saved here
        readnames = [] #A list saving all the read identifiers mapping to the respective
transcript- PRESUMABLY VERY MEMORY CONSUMING, SHOULD BE DEPRECATED
        lengths = [] #list of lenghts of all exons belonging to the transcript

        poscounter = 1 #defines the nucleotide position of the transcript a coverage value
is assigned to
        featureborders = [] #add the last poscounter value for every feature
        exonborders = [] #add the last poscounter value for every exon
        feature_readnums = [] #Stores the numbers of reads mapped to each feature of the
transcript
        feature_readlist = [] #Stores the names of the reads mapping to each feature of the
transcript  PRESUMABLY VERY MEMORY CONSUMING, SHOULD BE DEPRECATED
        feature_five_profile = []
        feature_three_profile = []

        if self.strand == "+":
            for lst in coordinates: #iterate over the lists of exon coordinates for the
transcripts features
                feature = []
                feature_reads=[]
                featurecount = 0

                for vals in lst: #Iterate over the coodinates saved in the current features
list
                    covdata = samfile.count_coverage(self.chromosome, vals[0]-1, vals[1])
#count the coverage within the range of the current exons start and stop position
                    lengths.append(vals[1]-vals[0]+1) # append the length of the curren exon
to lenghts
                    #covdata is a list of 4 lists containing the coverage of the exon split
up to the 4 nucleotides, IN FUTURE MAY ALLOW IDENTIFICATION OF SNPs
```

```
                    for i in range(len(covdata[0])): #iterate over the indexes of the lists
in covdata
                        '''
                        Here we use range(len(covdata[0])) to directly index all four
nucleotide coverages of the current exon and sum them up.
                        using poscounter, we build a dictionary that contains the summed
coverage for each position of the transcript. Poscounter is 1 in the beginning
                        and is incremented by 1 after each position that was calculated. In
this way, poscounter gives the correct 1-based indexed position of the base in the
                        mature transcript, regardless of how many different exons we have.
                        '''
                        coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]

                        poscounter +=1
                        featurecount += 1
                    feature.append(poscounter-1) #the last position of the exon is appended
to feature to track exon-exon borders later

                    for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]): #iterate
over all reads that align to the exon
                        '''
                        This part unfortunately is computationally very expensive, but
necessary to calculate rpkm or cpm. The BAM file is queried for all
                        reads that align to the current exon. Unfortunately the query also
yields spliced reads, whose splice junction just spans the queried region, but
                        do not map there. I tried the pysam.count() method, but it returned
the same error. Therefore the CIGAR of the fetched reads (which is already decoded)
                        has to be read to identify if the read really maps.
                        '''
                        currpos = read.pos #The current position to work on is the 5'-most
position of the read
                        matching_regions = []
                        for tup in read.cigar: #iterate over the tuples of the reads cigar
to decide what to do. First integer of tuple indicates cigar operation, secon the number of
bases affected.
                            if tup[0] in [0,7,8]: #if we have an alignment match, sequence
match or sequence mismatch
                                for pos in range(currpos,currpos+tup[1]): #iterate over the
positions in question (current position to current postion + the number of bases affected by
current operation)
                                    matching_regions.append(pos) #and append them to
matching_regions
                            elif tup[0] == [1,4,5,6]: #do nothing in these cases (insertion,
soft clipping, hard clipping, padding)
                                continue
                            else: # in every other case just increase the current position
by the number of bases affected by the current operation
                                currpos+=tup[1]
                        if any([(x >= vals[0]-1) & (x<= vals[1]) for x in
matching_regions]): #if any of the reads matching positions is located within the current
exon, then add that reads name to readnames and feature_reads
                            readnames.append(read.qname)
                            feature_reads.append(read.qname)

                featureborders.append(poscounter-1) #features last position to
featureborders
                exonborders.append(feature) #append the exonborders listed in feature to
exonboders
                feature_readnums.append(len(set(feature_reads))) #append the number of reads
mapped to the current feature to feature_readnums
                feature_readlist.append(set(feature_reads)) #append the names of read mapped
to the current feature to feature_readlist

        elif self.strand == "-": #do the same as above for - stranded transcripts

            for lst in coordinates:
                feature = []
                feature_reads = []

                for vals in lst:
                    covdata = samfile.count_coverage(self.chromosome, vals[0]-1, vals[1])
                    lengths.append(vals[1]-vals[0]+1)

                    for i in range(len(covdata[0])-1,-1,-1):
                        coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                        poscounter +=1
```

```
                    feature.append(poscounter-1)
                    for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]):
                        currpos = read.pos
                        matching_regions = []
                        for tup in read.cigar:
                            if tup[0] in [0,7,8]:
                                for pos in range(currpos,currpos+tup[1]):
                                    matching_regions.append(pos)
                            elif tup[0] == [1,4,5,6]:
                                continue
                            else:
                                currpos+=tup[1]
                        if any([(x >= vals[0]-1) & (x<= vals[1]) for x in
matching_regions]):
                            readnames.append(read.qname)
                            feature_reads.append(read.qname)

                featureborders.append(poscounter-1)
                exonborders.append(feature)
                feature_readnums.append(len(set(feature_reads)))
                feature_readlist.append(set(feature_reads))

        gbc = []
        threshold = 1
        if calc_gbc == True:
            for n in range(10):
                nonzerocov = 0
                for i in coverage:
                    if coverage[i] >= threshold:
                        nonzerocov += 1
                transcript_gbc = (nonzerocov/len(coverage))
                gbc.append(transcript_gbc)
                threshold += 1

        length = sum(lengths)
        reads = len(set(readnames))
        rpkm = reads/(length/1000)/self.mreads
        cpm = reads/self.mreads

        self.full_featureborders = featureborders
        self.full_exonborders = exonborders
        self.full_coverage = coverage
        if (get_readnames == True) or (self.name == "psbA"): #to save some memory the
readnames are discarded if not indicated otherwise
            self.full_readnames = set(readnames)
            self.full_feature_readlist = feature_readlist
        else:
            del feature_readlist
            del readnames
            self.full_readnames = None
            self.full_feature_readlist = None
        self.full_poscounter = poscounter
        self.full_gbc = gbc
        self.full_length = length
        self.full_reads = reads
        self.full_rpkm = rpkm
        self.full_cpm = cpm
        self.full_feature_readnums = feature_readnums

        return coverage, poscounter, gbc, length, reads, rpkm, cpm

    def only_cds(self, calc_gbc=False, get_readnames=False): #do the same as fulltranscript,
but ignore UTRs
        samfile = self.samfile
        if len(self.codings) == 0:
            coordinates = self.exons
        else:
            coordinates = self.codings
        coverage = {}
        readnames = []
        lengths = []
        poscounter = 1
        exonborders = []

        if self.strand == "+":

            for codingsequence in coordinates:
```

```
                covdata = samfile.count_coverage(self.chromosome, codingsequence[0]-1,
codingsequence[1])
                lengths.append(codingsequence[1]-codingsequence[0]+1)

                for i in range(len(covdata[0])):
                    coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                    poscounter +=1
                exonborders.append(poscounter-1)

                for read in samfile.fetch(self.chromosome, codingsequence[0]-1,
codingsequence[1]):
                    currpos = read.pos
                    matching_regions = []
                    for tup in read.cigar:
                        if tup[0] in [0,7,8]:
                            for pos in range(currpos,currpos+tup[1]):
                                matching_regions.append(pos)
                        elif tup[0] == [1,4,5,6]:
                            continue
                        else:
                            currpos+=tup[1]
                    if any([(x >= codingsequence[0]-1) & (x<= codingsequence[1]) for x in
matching_regions]):
                        readnames.append(read.qname)


        elif self.strand == "-":

            for codingsequence in coordinates:
                covdata = samfile.count_coverage(self.chromosome, codingsequence[0]-1,
codingsequence[1])
                lengths.append(codingsequence[1]-codingsequence[0]+1)

                for i in range(len(covdata[0])-1,-1,-1):
                    coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                    poscounter +=1
                exonborders.append(poscounter-1)

                for read in samfile.fetch(self.chromosome, codingsequence[0]-1,
codingsequence[1]):
                    currpos = read.pos
                    matching_regions = []
                    for tup in read.cigar:
                        if tup[0] in [0,7,8]:
                            for pos in range(currpos,currpos+tup[1]):
                                matching_regions.append(pos)
                        elif tup[0] == [1,4,5,6]:
                            continue
                        else:
                            currpos+=tup[1]
                    if any([(x >= codingsequence[0]-1) & (x<= codingsequence[1]) for x in
matching_regions]):
                        readnames.append(read.qname)


        gbc = []
        threshold = 1

        if calc_gbc == True:
            for n in range(10):
                nonzerocov = 0
                for i in coverage:
                    if coverage[i] >= threshold:
                        nonzerocov += 1
                transcript_gbc = (nonzerocov/len(coverage))
                gbc.append(transcript_gbc)
                threshold += 1

        length = sum(lengths)
        reads = len(set(readnames))
        rpkm = reads/(length/1000)/self.mreads
        cpm = reads/self.mreads

        self.cds_exonborders = exonborders
        self.cds_coverage = coverage
```

```
        if (get_readnames == True) or (self.name == "psbA"): #to save some memory the
readnames are discarded if not indicated otherwise
            self.cds_readnames = set(readnames)
        else:
            del readnames
            self.cds_readnames = None
        self.cds_poscounter = poscounter
        self.cds_gbc = gbc
        self.cds_length = length
        self.cds_reads = reads
        self.cds_rpkm = rpkm
        self.cds_cpm = cpm

        return coverage, poscounter, gbc, length, reads, rpkm, cpm

    def export_profile(self, mode="cds"):
        assert mode in ["cds", "full"], 'mode must be \'cds\' to export profile of coding
sequences or \'full\' for profile of the full transcript.'
        if mode == "full":
            try:
                exonborders = self.full_exonborders
                coverage = self.full_coverage
            except:
                print("self.full_exonborders does not exist. Run \".fulltranscript()\"
method and try again or use \"cds\" mode.")

            exonstart = 1 # correct - in cds mode exons don't start at 1 with
full_exoborders
            transcriptcov = []
            poscounter = 1
            for feature in exonborders:
                featurecov = []
                for coordinate in feature:
                    exoncov = []
                    positions = []
                    for x in range(exonstart,coordinate+1):
                        exoncov.append(coverage[x])
                        positions.append(poscounter)
                        poscounter += 1
                    exonstart = coordinate+1
                    featurecov.append([exoncov,positions])
                transcriptcov.append(featurecov)

            self.profile = (transcriptcov, "full")
            return (transcriptcov, "full")


        elif mode == "cds":
            try:
                exonborders = self.cds_exonborders
                coverage = self.cds_coverage
                startingpoint = 1
            except:
                print("\"self.cds_exonborders\" and \"self.cds_coverage\" do not exist,
trying to use \"self.full_exonborders\" and \"self.full_coverage\" instead.")
                try:
                    exonborders = self.full_exonborders[1]
                    coverage = self.full_coverage
                    startingpoint = self.full_exonborders[0][-1]+1
                except:
                    print("No exonborders or coverage data detected. Run
\".fulltranscript()\" or \".only_cds()\" method and try again.")

            cdscov = []
            exonstart = startingpoint
            poscounter = 1
            for exonend in exonborders:
                exoncov = []
                positions = []
                for x in range(exonstart,exonend+1):
                    exoncov.append(coverage[x])
                    positions.append(poscounter)
                    poscounter += 1
                cdscov.append([exoncov,positions])
                exonstart = exonend+1

            self.profile = (cdscov,"cds")
            return (cdscov, "cds")
```

```
        #-------SPECIAL FUNCTION FOR psaA ONLY!------------------

    def full_psaA(self):
        assert (self.identifier == "psaA_full"), "Method \"full_psaA\" is only allowed for
Transcript objects named \"psaA_full\"."

        coordinates = [self.five_utr, self.codings, self.three_utr]
        #all genomic parts are lists of tuples representing the exons of the part
        coverage = {}
        readnames = []
        lengths = []
        poscounter = 1 #defines the nucleotide position of the transcript a coverage value
is assigned to
        featureborders = [] #add the last poscounter value for every feature
        exonborders = [] #add the last poscounter value for every exon
        samfile = self.samfile

        feature = []
        for vals in self.five_utr:
            covdata = samfile.count_coverage(self.chromosome, vals[0]-1, vals[1])
            lengths.append(vals[1]-vals[0]+1)

            for i in range(len(covdata[0])):
                coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                poscounter +=1
            feature.append(poscounter-1)
            for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]):
                currpos = read.pos
                matching_regions = []
                for tup in read.cigar:
                    if tup[0] in [0,7,8]:
                        for pos in range(currpos,currpos+tup[1]):
                            matching_regions.append(pos)
                    elif tup[0] == [1,4,5,6]:
                        continue
                    else:
                        currpos+=tup[1]
                if any([(x >= vals[0]-1) & (x<= vals[1]) for x in matching_regions]):
                    readnames.append(read.qname)

            featureborders.append(poscounter-1)
            exonborders.append(feature)

        feature = []

        fragment = 1
        for lst in coordinates[1:3]:
            if fragment == 1:
                fragment +=1
                for vals in lst:
                    covdata = samfile.count_coverage(self.chromosome, vals[0]-1, vals[1])
                    lengths.append(vals[1]-vals[0]+1)

                    for i in range(len(covdata[0])):
                        coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                        poscounter +=1
                    feature.append(poscounter-1)
                    for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]):
                        currpos = read.pos
                        matching_regions = []
                        for tup in read.cigar:
                            if tup[0] in [0,7,8]:
                                for pos in range(currpos,currpos+tup[1]):
                                    matching_regions.append(pos)
                            elif tup[0] == [1,4,5,6]:
                                continue
                            else:
                                currpos+=tup[1]
                        if any([(x >= vals[0]-1) & (x<= vals[1]) for x in
matching_regions]):
                            readnames.append(read.qname)
                featureborders.append(poscounter-1)
                exonborders.append(feature)

            else:
```

```
                for vals in lst:
                    feature = []
                    for vals in lst:
                        covdata = samfile.count_coverage(self.chromosome, vals[0]-1,
vals[1])
                        lengths.append(vals[1]-vals[0]+1)

                        for i in range(len(covdata[0])-1,-1,-1):
                            coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                            poscounter +=1
                        feature.append(poscounter-1)
                        for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]):
                            currpos = read.pos
                            matching_regions = []
                            for tup in read.cigar:
                                if tup[0] in [0,7,8]:
                                    for pos in range(currpos,currpos+tup[1]):
                                        matching_regions.append(pos)
                                elif tup[0] == [1,4,5,6]:
                                    continue
                                else:
                                    currpos+=tup[1]
                            if any([(x >= vals[0]-1) & (x<= vals[1]) for x in
matching_regions]):
                                readnames.append(read.qname)

                    featureborders.append(poscounter-1)
                    exonborders.append(feature)

        gbc = []
        threshold = 1
        for n in range(10):
            nonzerocov = 0
            for i in coverage:
                if coverage[i] >= threshold:
                    nonzerocov += 1
            transcript_gbc = (nonzerocov/len(coverage))
            gbc.append(transcript_gbc)
            threshold += 1

        length = sum(lengths)
        reads = len(set(readnames))
        rpkm = reads/(length/1000)/self.mreads
        cpm = reads/self.mreads

        self.full_featureborders = featureborders
        self.full_exonborders = exonborders
        self.full_coverage = coverage
        self.full_readnames = readnames
        self.full_poscounter = poscounter
        self.full_gbc = gbc
        self.full_length = length
        self.full_reads = reads
        self.full_rpkm = rpkm
        self.full_cpm = cpm

        return coverage, readnames, poscounter, gbc, length, reads, rpkm, cpm


    def psaA_cds(self):
        assert (self.identifier == "psaA_cds"), "Method \"psaA_cds\" is only allowed for
Transcript objects named \"psaA_cds\"."

        coordinates = self.codings
        coverage = {}
        readnames = []
        lengths = []
        poscounter = 1
        exonborders = []
        samfile = self.samfile

        feature = []

        fragment = 1
        for vals in coordinates:
            if fragment == 1:
```

```
                fragment +=1
                if True:
                    covdata = samfile.count_coverage(self.chromosome, vals[0]-1, vals[1])
                    lengths.append(vals[1]-vals[0]+1)

                    for i in range(len(covdata[0])):
                        coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                        poscounter +=1
                    exonborders.append(poscounter-1)

                    for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]):
                        currpos = read.pos
                        matching_regions = []
                        for tup in read.cigar:
                            if tup[0] in [0,7,8]:
                                for pos in range(currpos,currpos+tup[1]):
                                    matching_regions.append(pos)
                            elif tup[0] == [1,4,5,6]:
                                continue
                            else:
                                currpos+=tup[1]
                        if any([(x >= vals[0]-1) & (x<= vals[1]) for x in
matching_regions]):
                            readnames.append(read.qname)

            else:
                if True:
                    covdata = samfile.count_coverage(self.chromosome, vals[0]-1, vals[1])
                    lengths.append(vals[1]-vals[0]+1)

                    for i in range(len(covdata[0])-1,-1,-1):
                        coverage[poscounter] =
covdata[0][i]+covdata[1][i]+covdata[2][i]+covdata[3][i]
                        poscounter +=1
                    exonborders.append(poscounter-1)

                    for read in samfile.fetch(self.chromosome, vals[0]-1, vals[1]):
                        currpos = read.pos
                        matching_regions = []
                        for tup in read.cigar:
                            if tup[0] in [0,7,8]:
                                for pos in range(currpos,currpos+tup[1]):
                                    matching_regions.append(pos)
                            elif tup[0] == [1,4,5,6]:
                                continue
                            else:
                                currpos+=tup[1]
                        if any([(x >= vals[0]-1) & (x<= vals[1]) for x in
matching_regions]):
                            readnames.append(read.qname)
        gbc = []
        threshold = 1
        for n in range(10):
            nonzerocov = 0
            for i in coverage:
                if coverage[i] >= threshold:
                    nonzerocov += 1
            transcript_gbc = (nonzerocov/len(coverage))
            gbc.append(transcript_gbc)
            threshold += 1

        length = sum(lengths)
        reads = len(set(readnames))
        rpkm = reads/(length/1000)/self.mreads
        cpm = reads/self.mreads

        self.cds_exonborders = exonborders
        self.cds_coverage = coverage
        self.cds_readnames = set(readnames)
        self.cds_poscounter = poscounter
        self.cds_gbc = gbc
        self.cds_length = length
        self.cds_reads = reads
        self.cds_rpkm = rpkm
        self.cds_cpm = cpm
    def psite_profile(self, fiveoffsets):
        chromosome = self.chromosome
```

```
        samfile = self.samfile
        frames=[]
        strand = self.strand
        for exon in self.codings:
            start = min(exon)
            stop = max(exon)
            startpos = start
            stoppos = stop+1
            psites=[]
            psite_lengths = {}
            for i in samfile.fetch(chromosome, startpos, stoppos):
                pos = i.pos
                currentpos = pos
                #length = len(i.seq)
                length = sum([x[1] for x in i.cigar if x[0] in [0,1,7,8]])
                try:
                    offset = abs(fiveoffsets[length])
                except KeyError:
                    continue
                remaining = offset
                lastpos = pos+sum([x[1] for x in i.cigar if x[0] in [0,2,3,7]])-1
                #if first cigar operation is M and operation range is greater than the
offset, OR if first cigar OP is Hard/Soft masked and the second is match + greater than
offset.
                if ((i.cigar[0][0] == 0) & (i.cigar[0][1] >= offset+1)):
                    if strand == "+":
                        psite = i.pos+offset
                    elif strand == "-":
                        psite = lastpos-offset
                elif (((i.cigar[0][0]==4) | (i.cigar[0][0]== 5)) & ((i.cigar[1][0] == 0) &
(i.cigar[1][1] >= length+1))):
                    if strand == "+":
                        psite = i.pos+offset
                    elif strand == "-":
                        psite = lastpos-offset
                else:
                    for c in i.cigar:
                        if c[0] in [1,4,5,6]:
                            continue
                        elif c[0] in [0,7,8]:
                            if c[1] < remaining:
                                if strand == "+":
                                    remaining -= c[1]
                                    currentpos += c[1]
                                elif strand == "-":
                                    remaining -= c[1]
                                    currentpos -= c[1]
                            elif c[1] >= remaining:
                                if strand == "+":
                                    psite = currentpos+remaining
                                elif strand == "-":
                                    psite = currentpos-remaining
                        elif c[0] in [2,3]:
                            if strand == "+":
                                currentpos += c[1]
                            elif strand == "-":
                                currentpos -= c[1]
                psites.append(psite)
                psite_lengths[psite] = psite_lengths.setdefault(psite,
[])+[i.query_alignment_length]

            if strand == "-":
                ind = np.flip(np.arange(startpos, stoppos))
            elif strand == "+":
                ind = np.arange(startpos, stoppos)
            p_counts = np.unique(psites, return_counts=True)
            p_table = pd.DataFrame(np.zeros(shape=len(ind)), index=ind, columns=["counts"])
            p_table["counts"] = pd.Series(p_counts[1], index=p_counts[0])
            p_table = p_table.fillna(0)
            frames.append(p_table)


        p_table = pd.concat(frames)
        if strand == "+":
            p_table.sort_index(inplace=True)
        elif strand == "-":
            p_table.sort_index(inplace=True, ascending=False)
```

```
        p_table["pos"] = list(range(len(p_table)))
        p_table["codon"] = p_table.apply(lambda row: int(row["pos"]/3)+1,axis=1)
        p_table["codsum"] = p_table.apply(lambda row:
p_table[p_table["codon"]==row["codon"]]["counts"].sum(), axis=1)
        p_table["pref"] = p_table.apply(lambda row: row["counts"]/row["codsum"] if
row["codsum"] != 0 else 0, axis=1)
        p_table["avg_rlength"] = p_table.apply(lambda row: np.mean(psite_lengths[row.name])
if row.name in psite_lengths.keys() else 0, axis=1)
        p_table["sd_rlength"] = p_table.apply(lambda row: np.std(psite_lengths[row.name]) if
row.name in psite_lengths.keys() else 0, axis=1)
        p_table["median_rlength"] = p_table.apply(lambda row:
np.median(psite_lengths[row.name]) if row.name in psite_lengths.keys() else 0, axis=1)
        p_table["frame"] = p_table.apply(lambda row: row["pos"]%3, axis=1)

        self.p_table = p_table
        return p_table


    def periodicity(self, fiveoffsets, profilesize=(30,30)):
        firstpatch = profilesize[0]*3
        lastpatch = profilesize[1]*3
        lengths = []
        chromosome = self.chromosome
        samfile = self.samfile
        strand = self.strand

        for codingsequence in self.codings:
            lengths.append(codingsequence[1]-codingsequence[0]+1)
        length = sum(lengths)

        if length < firstpatch+lastpatch:
            print("transcript "+self.name+" too short for periodicity calculation. Skip to
next transcript.")
            self.periodicity_table =
pd.DataFrame(np.zeros(shape=((firstpatch+lastpatch),5)), columns=["counts", "pos", "codon",
"codsum", "pref"])
            return

        codingpositions = []
        for exon in self.codings:
            start = min(exon)
            stop = max(exon)
            startpos = start-1
            stoppos = stop
            positions = np.arange(startpos, stoppos)
            codingpositions.append(positions)
        codingpositions = np.concatenate(codingpositions)
        if strand == "-":
            codingpositions = -np.sort(-codingpositions)
        elif strand == "+":
            codingspositions = np.sort(codingpositions)

        firstpositions = codingpositions[0:firstpatch]
        lastpositions = codingpositions[-lastpatch:]
        self.firstpositions = firstpositions
        self.lastpositions = lastpositions

        def subsequences(target, strand):
            postuples = []
            mode=0
            for x in target:
                if mode == 0:
                    start = x
                    t = x
                    mode+=1
                elif mode != 0:
                    if (x == target[-1])&(abs(x-t)>1):
                        if strand == "-":
                            stop = t
                            start= start+1
                            postuples.append(tuple(sorted([start,stop])))
                            stop = x
                            start= x+1
                        elif strand == "+":
                            stop = t+1
                            postuples.append(tuple(sorted([start,stop])))
                            start = x
                            stop = x+1
```

```
                        postuples.append(tuple(sorted([start,stop])))

                    elif abs(x-t) > 1:
                        if strand == "-":
                            stop = t
                            start= start+1
                        elif strand == "+":
                            stop = t+1
                        postuples.append(tuple(sorted([start,stop])))
                        start = x
                        t = x

                    elif x == target[-1]:
                        if strand == "-":
                            stop = x
                            start= start+1
                        elif strand == "+":
                            stop = x+1
                        postuples.append(tuple(sorted([start,stop])))

                    elif abs(x-t) == 1:
                        t=x

            return postuples
        firsttuples = subsequences(target=firstpositions, strand= self.strand)
        lasttuples = subsequences(target= lastpositions, strand = self.strand)
        self.firsttuples = firsttuples
        self.lasttuples = lasttuples

        frames = []
        for tup in firsttuples:
            start = min(tup)
            stop = max(tup)
            psites=[]

            for i in samfile.fetch(chromosome, start, stop):
                pos = i.pos
                currentpos = pos
                #length = len(i.seq)
                length = sum([x[1] for x in i.cigar if x[0] in [0,1,7,8]])
                try:
                    offset = abs(fiveoffsets[length])
                except KeyError:
                    continue
                remaining = offset
                lastpos = pos+sum([x[1] for x in i.cigar if x[0] in [0,2,3,7]])-1
                #if first cigar operation is M and operation range is greater than the
offset, OR if first cigar OP is Hard/Soft masked and the second is match + greater than
offset.
                if ((i.cigar[0][0] == 0) & (i.cigar[0][1] >= offset+1)):
                    if strand == "+":
                        psite = i.pos+offset
                    elif strand == "-":
                        psite = lastpos-offset
                elif (((i.cigar[0][0]==4) | (i.cigar[0][0]== 5)) & ((i.cigar[1][0] == 0) &
(i.cigar[1][1] >= length+1))):
                    if strand == "+":
                        psite = i.pos+offset
                    elif strand == "-":
                        psite = lastpos-offset
                else:
                    for c in i.cigar:
                        if c[0] in [1,4,5,6]:
                            continue
                        elif c[0] in [0,7,8]:
                            if c[1] < remaining:
                                if strand == "+":
                                    remaining -= c[1]
                                    currentpos += c[1]
                                elif strand == "-":
                                    remaining -= c[1]
                                    currentpos -= c[1]
                            elif c[1] >= remaining:
                                if strand == "+":
                                    psite = currentpos+remaining
                                elif strand == "-":
                                    psite = currentpos-remaining
                        elif c[0] in [2,3]:
```

```
                            if strand == "+":
                                currentpos += c[1]
                            elif strand == "-":
                                currentpos -= c[1]
                    psites.append(psite)

                if strand == "-":
                    ind = np.flip(np.arange(start, stop))
                elif strand == "+":
                    ind = np.arange(start, stop)

                p_counts = np.unique(psites, return_counts=True)
                p_table = pd.DataFrame(np.zeros(shape=len(ind)), index=ind, columns=["counts"])
                p_table["counts"] = pd.Series(p_counts[1], index=p_counts[0])
                p_table = p_table.fillna(0)
                frames.append(p_table)
            if len(frames) > 1:
                p_table = pd.concat(frames)
            else:
                p_table = frames[0]
            if strand == "+":
                p_table.sort_index(inplace=True)
            elif strand == "-":
                p_table.sort_index(inplace=True, ascending=False)

            self.firstperiodicity = p_table

            frames=[]
            for tup in lasttuples:
                start = min(tup)
                stop = max(tup)
                psites=[]

                for i in samfile.fetch(chromosome, start, stop):
                    pos = i.pos
                    currentpos = pos
                    #length = len(i.seq)
                    length = sum([x[1] for x in i.cigar if x[0] in [0,1,7,8]])
                    try:
                        offset = abs(fiveoffsets[length])
                    except KeyError:
                        continue
                    remaining = offset
                    lastpos = pos+sum([x[1] for x in i.cigar if x[0] in [0,2,3,7]])-1
                    #if first cigar operation is M and operation range is greater than the
offset, OR if first cigar OP is Hard/Soft masked and the second is match + greater than
offset.
                    if ((i.cigar[0][0] == 0) & (i.cigar[0][1] >= offset+1)):
                        if strand == "+":
                            psite = i.pos+offset
                        elif strand == "-":
                            psite = lastpos-offset
                    elif (((i.cigar[0][0]==4) | (i.cigar[0][0]== 5)) & ((i.cigar[1][0] == 0) &
(i.cigar[1][1] >= length+1))):
                        if strand == "+":
                            psite = i.pos+offset
                        elif strand == "-":
                            psite = lastpos-offset
                    else:
                        for c in i.cigar:
                            if c[0] in [1,4,5,6]:
                                continue
                            elif c[0] in [0,7,8]:
                                if c[1] < remaining:
                                    if strand == "+":
                                        remaining -= c[1]
                                        currentpos += c[1]
                                    elif strand == "-":
                                        remaining -= c[1]
                                        currentpos -= c[1]
                                elif c[1] >= remaining:
                                    if strand == "+":
                                        psite = currentpos+remaining
                                    elif strand == "-":
                                        psite = currentpos-remaining
                            elif c[0] in [2,3]:
                                if strand == "+":
                                    currentpos += c[1]
```

```
                            elif strand == "-":
                                currentpos -= c[1]
                    psites.append(psite)

                if strand == "-":
                    ind = np.flip(np.arange(start, stop))
                elif strand == "+":
                    ind = np.arange(start, stop)
                p_counts = np.unique(psites, return_counts=True)
                p_table = pd.DataFrame(np.zeros(shape=len(ind)), index=ind, columns=["counts"])
                p_table["counts"] = pd.Series(p_counts[1], index=p_counts[0])
                p_table = p_table.fillna(0)
                frames.append(p_table)
            if len(frames) > 1:
                p_table = pd.concat(frames)
            elif len(frames) == 1:
                p_table = frames[0]
            if strand == "+":
                p_table.sort_index(inplace=True)
            elif strand == "-":
                p_table.sort_index(inplace=True, ascending=False)

            self.lastperiodicity = p_table
            periodicity_table = pd.concat([self.firstperiodicity, self.lastperiodicity],
ignore_index=True).reset_index()
            periodicity_table["pos"] = list(range(len(periodicity_table)))
            periodicity_table["codon"] = periodicity_table.apply(lambda row:
int(row["pos"]/3)+1,axis=1)
            periodicity_table["codsum"] = periodicity_table.apply(lambda row:
periodicity_table[periodicity_table["codon"]==row["codon"]]["counts"].sum(), axis=1)
            periodicity_table["pref"] = periodicity_table.apply(lambda row:
row["counts"]/row["codsum"] if row["codsum"] != 0 else 0, axis=1)

            self.periodicity_table = periodicity_table
            return

    def BuildTranscripts(genomic_coordinates, samfile, chlamy=True):
        samfile = samfile
        transcript_objects = {}
        counter = 1
        ts = time()
        for index,i in genomic_coordinates.iterrows():
            if counter in [100,500]:
                print("%s transcripts created, took %s seconds. Makes %s seconds per object."
%(counter, time()-ts, (time()-ts)/counter))
            elif (counter < 60000)&(counter%10000 == 0):
                print("%s transcripts created, took %s seconds. Makes %s seconds per object."
%(counter, time()-ts, (time()-ts)/counter))
            elif counter%25000 == 0:
                print("%s transcripts created, took %s seconds. Makes %s seconds per object."
%(counter, time()-ts, (time()-ts)/counter))
            try:
                identifier = i["identifier"]
                chromosome = i["chromosome"]
                strand = i["strand"]
                five_utr = i["five_prime_UTR"]
                three_utr = i["three_prime_UTR"]
                codings = i["codings"]
                name = i["name"]
                info = i["info"]
                exons = i["exons"]
                transcript_objects[identifier]= Transcript(chromosome=chromosome, strand=strand,
five_utr=five_utr, three_utr= three_utr, codings=codings, exons=exons, name= name,
identifier= identifier, samfile=samfile, info=info)
                counter += 1
            except IndexError:
                print(counter)
                break
        if chlamy==True:
            print("All transcripts created, now  preparing for psbA and psaA special rules")
            transcript_objects["CreCp.g802280_4532.1.v6.1"].fulltranscript()
            transcript_objects["CreCp.g802281_4532.1.v6.1"].fulltranscript()
            transcript_objects["CreCp.g802282_4532.1.v6.1"].fulltranscript()
            transcript_objects["CreCp.g802285_4532.1.v6.1"].fulltranscript()
            transcript_objects["CreCp.g802321_4532.1.v6.1"].fulltranscript()
            transcript_objects["CreCp.g802280_4532.1.v6.1"].only_cds()
            transcript_objects["CreCp.g802281_4532.1.v6.1"].only_cds()
            transcript_objects["CreCp.g802282_4532.1.v6.1"].only_cds()
```

```
        transcript_objects["CreCp.g802285_4532.1.v6.1"].only_cds()
        transcript_objects["CreCp.g802321_4532.1.v6.1"].only_cds()
        print("Calculating psbA")
        combine_psbA_copies(how="full", transcript_objects=transcript_objects)
        combine_psbA_copies(how="cds", transcript_objects=transcript_objects)
        print("Calculating psaA")
        psaA_five_utr = transcript_objects["CreCp.g802280_4532.1.v6.1"].five_utr
        psaA_codings =
transcript_objects["CreCp.g802280_4532.1.v6.1"].codings+transcript_objects["CreCp.g802281_45
32.1.v6.1"].codings+transcript_objects["CreCp.g802282_4532.1.v6.1"].codings
        psaA_exons =
transcript_objects["CreCp.g802280_4532.1.v6.1"].exons+transcript_objects["CreCp.g802281_4532
.1.v6.1"].exons+transcript_objects["CreCp.g802282_4532.1.v6.1"].exons
        psaA_three_utr = transcript_objects["CreCp.g802282_4532.1.v6.1"].three_utr
        transcript_objects["psaA_full"] = Transcript(chromosome="plastome", strand="x",
five_utr=psaA_five_utr, three_utr= psaA_three_utr, codings=psaA_codings, exons=psaA_exons
,name= "psaA_full", identifier= "psaA_full", info=None, sort=False, samfile=samfile)
        transcript_objects["psaA_cds"] = Transcript(chromosome="plastome", strand="x",
five_utr=psaA_five_utr, three_utr= psaA_three_utr, codings=psaA_codings, exons=psaA_exons,
name= "psaA_cds", identifier= "psaA_cds", info=None, sort=False, samfile=samfile)
        transcript_objects["psaA_cds"].psaA_cds()
        transcript_objects["psaA_full"].full_psaA()
    else:
        print("Done!")
    return transcript_objects



def combine_psbA_copies(transcript_objects, how="auto"):
    first = transcript_objects["CreCp.g802285_4532.1.v6.1"]
    other = transcript_objects["CreCp.g802321_4532.1.v6.1"]

    if how == "auto":
        try:
            first_cdsborders = [first.full_exonborders[0][-1]]+first.full_exonborders[1]
            other_cdsborders = [other.full_exonborders[0][-1]]+other.full_exonborders[1]
            first_cov = first.full_coverage
            other_cov = other.full_coverage
            mode = "full"
        except:
            first_cdsborders = first.cds_exonborders
            other_cdsborders = other.cds_exonborders
            first_cov = first.cds_coverage
            other_cov = other.cds_coverage
            mode = "cds"

    elif how == "full":
        try:
            first_cdsborders = [first.full_exonborders[0][-1]]+first.full_exonborders[1]
            other_cdsborders = [other.full_exonborders[0][-1]]+other.full_exonborders[1]
            first_cov = first.full_coverage
            other_cov = other.full_coverage
            mode = "full"
        except:
            print("ERROR! Cannot combine psbA transcripts with \"how=\"full\" \"")
    elif how == "cds":
        try:
            first_cdsborders = first.cds_exonborders
            other_cdsborders = other.cds_exonborders
            first_cov = first.cds_coverage
            other_cov = other.cds_coverage
            mode = "cds"
        except:
            print("ERROR! Cannot combine psbA transcripts with \"how=\"cds\" \"")

    first_newcov = []
    other_newcov = []

    for i in range(first_cdsborders[0],(first_cdsborders[-1])+1):
        first_newcov.append(first_cov[i])
    for i in range(other_cdsborders[0],(other_cdsborders[-1])+1):
        other_newcov.append(other_cov[i])
    combined_cov = []
    for i in zip(first_newcov,other_newcov):
        combined_cov.append(sum(i))
```

```
    if mode == "full":
        for i,n in zip(range(first_cdsborders[0],first_cdsborders[-1]+1),combined_cov):
            transcript_objects["CreCp.g802285_4532.1.v6.1"].full_coverage[i] = n
        for i,n in zip(range(other_cdsborders[0],other_cdsborders[-1]+1),combined_cov):
            transcript_objects["CreCp.g802321_4532.1.v6.1"].full_coverage[i] = n
        combined_reads = first.full_feature_readnums[1]+other.full_feature_readnums[1]

        for transcript in
[transcript_objects["CreCp.g802285_4532.1.v6.1"],transcript_objects["CreCp.g802321_4532.1.v6
.1"]]:
            gbc = []
            threshold = 1
            coverage = transcript.full_coverage
            for n in range(10):
                nonzerocov = 0
                for i in coverage:
                    if coverage[i] >= threshold:
                        nonzerocov += 1
                transcript_gbc = (nonzerocov/len(coverage))
                gbc.append(transcript_gbc)
                threshold += 1


            reads =
transcript.full_feature_readnums[0]+combined_reads+transcript.full_feature_readnums[1]
            rpkm = reads/(transcript.full_length/1000)/transcript.mreads
            cpm = reads/transcript.mreads
            cds_reads =
set(list(transcript_objects["CreCp.g802285_4532.1.v6.1"].full_feature_readlist[1])+list(tran
script_objects["CreCp.g802321_4532.1.v6.1"].full_feature_readlist[1]))
            newreadlist = [transcript.full_feature_readlist[0], cds_reads,
transcript.full_feature_readlist[2]]


            setattr(transcript,"full_readnames",
set(list(transcript.full_readnames)+list(cds_reads)))
            setattr(transcript,"full_gbc", gbc)
            setattr(transcript,"full_reads", reads)
            setattr(transcript,"full_rpkm", rpkm)
            setattr(transcript,"full_cpm", cpm)
            setattr(transcript,"full_feature_readlist", newreadlist)
            setattr(transcript,"full_feature_readnums",
[len(transcript.full_feature_readlist[0]),len(transcript.full_feature_readlist[1]),len(trans
cript.full_feature_readlist[2])])

    else:
        for i,n in zip(range(1,first_cdsborders[-1]+1),combined_cov):
            transcript_objects["CreCp.g802285_4532.1.v6.1"].cds_coverage[i] = n
        for i,n in zip(range(1,other_cdsborders[-1]+1),combined_cov):
            transcript_objects["CreCp.g802321_4532.1.v6.1"].cds_coverage[i] = n
        combined_reads = first.cds_reads+other.cds_reads

        for transcript in
[transcript_objects["CreCp.g802285_4532.1.v6.1"],transcript_objects["CreCp.g802321_4532.1.v6
.1"]]:
            gbc = []
            threshold = 1
            coverage = transcript.cds_coverage
            for n in range(10):
                nonzerocov = 0
                for i in coverage:
                    if coverage[i] >= threshold:
                        nonzerocov += 1
                transcript_gbc = (nonzerocov/len(coverage))
                gbc.append(transcript_gbc)
                threshold += 1



            rpkm =combined_reads/(transcript.cds_length/1000)/transcript.mreads
            cpm = combined_reads/transcript.mreads


setattr(transcript,"cds_readnames",set(list(transcript_objects["CreCp.g802285_4532.1.v6.1"].
cds_readnames)+list(transcript_objects["CreCp.g802321_4532.1.v6.1"].cds_readnames)))
            setattr(transcript, "cds_gbc", gbc)
            setattr(transcript, "cds_reads", combined_reads)
            setattr(transcript, "cds_rpkm", rpkm)
```

```
            setattr(transcript, "cds_cpm", cpm)


def plot_coverage(transcript, save=False, savepath="", show=True, scale=False,
scaleto=10000000):
    cov = transcript.profile[0]

    fig1 = plt.figure()
    if scale == True:
        scalefactor = scaleto / transcript.totreads
        newfeatures=[[],[],[]]
        for feature, counter in zip(cov,[0,1,2]):
            for exon in feature:
                scaled = [float(n)*scalefactor for n in exon[0]]
                newexon = [scaled, exon[1]]
                newfeatures[counter].append(newexon)
        cov = newfeatures
    if transcript.profile[1] == "full":
        count=0
        for i in cov[0]: #plot 5'UTR
            coverage = i[0]
            positions= i[1]
            if count%2==0:
                dye = "#F9240F" #dark red
            else:
                dye = "#FC8D82" #light red
            plt.bar(x=positions,height=coverage, color=dye, width=1)
            count += 1

        count=0
        for i in cov[1]:      #plot CDSs
            coverage = i[0]
            positions= i[1]
            if count%2==0:
                dye = "#0B90FF" #dark blue
            else:
                dye = "#80C5FF" #light blue
            plt.bar(x=positions,height=coverage, color=dye, width=1)
            count += 1

        count=0
        for i in cov[2]:        #plot 3'UTRs
            coverage = i[0]
            positions= i[1]
            if count%2==0:
                dye = "#5CD30B"    #dark green
            else:
                dye = "#B4ED8D"     #light green
            plt.bar(x=positions,height=coverage, color=dye, width=1)
            count += 1
        #plt.set_xticks()
        plt.xlabel("position from TSS [nt]")
        cds_start = min(cov[1][0][1])
        cds_stop = max(cov[1][-1][1])
        plt.annotate("", xy=(cds_start,0), xytext=(cds_start,-(plt.axis()[3]/200)),
arrowprops= dict(facecolor="black", headwidth=5, headlength=6))
        plt.annotate("", xy=(cds_stop,0), xytext=(cds_stop,-(plt.axis()[3]/200)),
arrowprops= dict(facecolor="black", headwidth=5, headlength=6))

    elif transcript.profile[1] == "cds":
        count=0
        for i in cov:      #plot CDSs
            coverage = i[0]
            positions= i[1]
            if count%2==0:
                dye = "#0B90FF" #dark blue
            else:
                dye = "#80C5FF" #light blue
            plt.bar(x=positions,height=coverage, color=dye, width=1)
            count += 1
        plt.xlabel("position from start codon [nt]")
    figuretitle = str(transcript.name)+"\n("+str(transcript.identifier)+")"
    identifier = str(transcript.identifier)
    if figuretitle != "None":
        plt.title(label =figuretitle)
    else:
        plt.title(label = identifier)
```

```
    plt.ylabel("coverage [reads]")
    if save == True:
        assert (savepath != ""), "You have to pass a valid path to plot_coverage\'s
\"savepath\" parameter."
        plt.savefig(savepath)
    #fig1 = plt.gcf()
    if show==True:
        fig1.show()

    return fig1



def export_basic_information(transcript_objects, mode="cds", gene_body_coverage=False):
    exceptions =
["CreCp.g802285_4532.1.v6.1","CreCp.g802321_4532.1.v6.1","psaA_full","psaA_cds"]
    results = []
    counter = 1
    ts = time()
    for i in transcript_objects:
        if counter in [100,500]:
            print("%s transcripts calculated, took %s seconds. Makes %s seconds per object."
%(counter, time()-ts, (time()-ts)/counter))
        elif (counter < 60000)&(counter%10000 == 0):
            print("%s transcripts calculated, took %s seconds. Makes %s seconds per object."
%(counter, time()-ts, (time()-ts)/counter))
        elif counter%25000 == 0:
            print("%s transcripts calculated, took %s seconds. Makes %s seconds per object."
%(counter, time()-ts, (time()-ts)/counter))
        x = transcript_objects[i]
        if x.identifier in exceptions:
            if mode == "cds":
                try:
                    entry = [x.identifier, x.name, x.chromosome, x.strand, x.cds_gbc,
x.cds_length, x.cds_reads, x.cds_rpkm, x.cds_cpm]
                    results.append(entry)
                    continue
                except AttributeError:
                    entry = [x.identifier, x.name, x.chromosome, x.strand, x.full_gbc,
x.full_length, x.full_reads, x.full_rpkm, x.full_cpm]
                    results.append(entry)
                    continue
            elif mode == "full":
                try:
                    entry = [x.identifier, x.name, x.chromosome, x.strand, x.full_gbc,
x.full_length, x.full_reads, x.full_rpkm, x.full_cpm]
                    results.append(entry)
                    continue
                except AttributeError:
                    entry = [x.identifier, x.name, x.chromosome, x.strand, x.cds_gbc,
x.cds_length, x.cds_reads, x.cds_rpkm, x.cds_cpm]
                    results.append(entry)
                    continue
        if mode == "cds":
            x.only_cds(calc_gbc=gene_body_coverage)
            entry = [x.identifier, x.name, x.chromosome, x.strand, x.cds_gbc, x.cds_length,
x.cds_reads, x.cds_rpkm, x.cds_cpm, x.info]
        elif mode == "full":
            x.fulltranscript(calc_gbc=gene_body_coverage)
            entry = [x.identifier, x.name, x.chromosome, x.strand, x.full_gbc,
x.full_length, x.full_reads, x.full_rpkm, x.full_cpm, x.info]
        #y = x.only_cds()
        #occu.append(y[3::])
        results.append(entry)
        counter += 1
    print("All done. Took %s seconds. Makes %s seconds per object." %(time()-ts, (time()-
ts)/counter))
    fulllist = []
    for i in results:
        new = []
        for x in i:
            if type(x) == list:
                if len(x)==0:
                    continue
                elif type(x[0])!=float:
                    new.append(x)
                else:
                    for y in x:
```

```
                        new.append(y)
            else:
                new.append(x)
        fulllist.append(new)
    if gene_body_coverage == True:
        summary = pd.DataFrame(fulllist,
columns=["identifier","name","chromosome","strand","1","2","3","4","5","6","7","8","9","10",
"length","reads","rpkm","cpm","info"])
    elif gene_body_coverage == False:
        summary = pd.DataFrame(fulllist,
columns=["identifier","name","chromosome","strand","length","reads","rpkm","cpm","info"])

    return summary


def prepare_genomic_coordinates(path_to_annotation, delimiter, skiprows=0, comment="#",
parenttag="Parent=", idtag="ID=", nametag="Name=", genetag="geneName=",
transcripttag="mRNA", organism="Chlamydomonas"):

    def splitter(row, x, y):
        counter= 0
        output=None
        keyword = x
        secondkeyword = y
        try:
            output = row[8].split(x)[1].split(";")[0]
        except IndexError:
            try:
                if organism == "human":
                    output = row[2]+":"+row[8].split(y)[1].split(";")[0] #for hg38
                elif organism == "Chlamydomonas":
                    output = (row[8].split(y)[1].split(";")[0])+"."+row[2] #for chlamy v6
            except IndexError:
                output=None
        return output

    annotation = pd.read_csv(path_to_annotation, delimiter=delimiter, skiprows=skiprows,
names=[0,1,2,3,4,5,6,7,8], comment=comment)
    annotation2 = annotation.copy(deep=True)
    annotation2["ID"]= annotation.apply(lambda row: splitter(row, idtag, parenttag), axis=1)
    annotation2["Parent"]= annotation.apply(lambda row: splitter(row, parenttag, nametag),
axis=1)
    annotation2["coordinates"] = annotation.apply(lambda row: tuple([row[3],row[4]]),
axis=1)

    print("IDs and Parents annotated")

    results = {}
    ts = time()
    print("start at ", time())
    counter = 1
    if organism == "Chlamydomonas":
        candidates = annotation2[(annotation2[2]== transcripttag)]
    elif organism == "human":
        candidates = annotation2[(annotation2[2]==
transcripttag)&(annotation2[8].str.contains("gene_type=protein_coding"))] #for hg38 to
remove all non coding transcripts from the searchspace
    for i in candidates.itertuples(name=None):
        if counter == 100:
            print("100 loops done. Took ", time()-ts)
        elif counter == 500:
            print("500 loops done. Took ", time()-ts)
        elif counter%1000==0:
            print(counter, " loops done. Took ", time()-ts)


        transcript = i
        cds_coordinates = []
        fiveprime_coordinates = []
        threeprime_coordinates = []
        exon_coordinates = []
        strand = transcript[7]
        info=transcript[9]
        chromosome = transcript[1]

        searchphrase = transcript[10]
        if genetag in str(transcript[9]):
            name = str(transcript[9]).split(genetag)[1].split(";")[0]
```

```
        else:
            name = searchphrase


        #searchspace =
annotation2[(annotation2[0]==i[1])&(annotation2[3]>=i[4])&(annotation2[4]<=i[5])]
        searchspace = annotation2[annotation2["Parent"] == searchphrase]

        fiveprime_coordinates = searchspace[(searchspace[2]==
"five_prime_UTR")&(searchspace["Parent"]== searchphrase)]["coordinates"].tolist()
        threeprime_coordinates = searchspace[(searchspace[2]==
"three_prime_UTR")&(searchspace["Parent"]== searchphrase)]["coordinates"].tolist()
        cds_coordinates = searchspace[(searchspace[2]== "CDS")&(searchspace["Parent"]==
searchphrase)]["coordinates"].tolist()
        exon_coordinates = searchspace[(searchspace[2]== "exon")&(searchspace["Parent"]==
searchphrase)]["coordinates"].tolist()
        '''
        for row in searchspace[(searchspace[2]== "five_prime_UTR")&(searchspace["Parent"]==
searchphrase)].itertuples(name=None):
            fiveprime_coordinates.append((row[4],row[5]))

        for row in searchspace[(searchspace[2]== "three_prime_UTR")&(searchspace["Parent"]==
searchphrase)].itertuples(name=None):
            threeprime_coordinates.append((row[4],row[5]))

        for row in searchspace[(searchspace[2]== "CDS")&(searchspace["Parent"]==
searchphrase)].itertuples(name=None):
            cds_coordinates.append((row[4],row[5]))

        for row in searchspace[(searchspace[2]== "exon")&(searchspace["Parent"]==
searchphrase)].itertuples(name=None):
            exon_coordinates.append((row[4],row[5]))
        '''
        coordinates = [chromosome, strand, fiveprime_coordinates, threeprime_coordinates,
cds_coordinates, exon_coordinates, name, searchphrase, info]

        results[searchphrase]= coordinates
        counter += 1

    print("All done. Took ", time()-ts, (". Makes "), (time()-ts)/len(results), " per
loop.")
    genomic_coordinates = pd.DataFrame.from_dict(results, orient="index",
columns=["chromosome","strand","five_prime_UTR","three_prime_UTR","codings","exons","name","
identifier","info"])
    pickle_name = path_to_annotation+"_coordinates.pkl"
    genomic_coordinates.to_pickle(pickle_name)

    return genomic_coordinates

def multiplot(identifier, path, datasets, data_set_names="undefined", fmt="pdf"): #just
annotate the identifier, the savepath and the list of datasets (as dictionaries of
transcript objects)to comapre, currently only possible in cds mode
    if data_set_names == "undefined":
        data_set_names = list([str(i) for i in range(len(datasets))])
    base = datasets[0]
    transcript = base[identifier]
    identifier = transcript.identifier
    name = transcript.name
    if fmt=="pdf":
        suffix=".pdf"
    elif fmt=="png":
        suffix=".png"
    if name == "None":
        name = identifier
    path = path+name+suffix
    newcov = []
    plt.clf()
    plt.style.use("seaborn-talk")
    for data in datasets:
        try:
            cov = data[identifier].cds_coverage
            reads = data[identifier].cds_reads
            total = sum(data[identifier].cds_coverage.values())

        except:
            data[identifier].only_cds()
            #data[identifier].export_profile()
            cov = data[identifier].cds_coverage
```

```
            reads = data[identifier].cds_reads
            total = sum(data[identifier].cds_coverage.values())
        if total == 0:
            ncov=list(len(cov)*[0])
        else:
            ncov=list(map(lambda x: x/total*100,cov.values()))
        newcov.append(ncov)
    plt.figure(figsize=(10.0,5.0))
    for i,n in zip(newcov,data_set_names):
        plt.plot(i, label=n)
    plt.legend()
    plt.ylabel("normalized coverage [%]")
    plt.xlabel("position from start codon [nt]")
    figuretitle = str(transcript.name)+"\n("+str(transcript.identifier)+")"
    identifier = str(transcript.identifier)
    if name != "None":
        plt.title(label =figuretitle)
    else:
        plt.title(label = identifier)
    plt.savefig(path, format=fmt)
    plt.show()

def offset_calc(dataset, five_offset = True, three_offset = True, readlengths = (20,39),
min_five=20, min_three=20, chloroplast="plastome", mitochondria="mitogenome"):
    '''
    calculate p-site offset. From each transcript extract all reads mapping onto the A of
start codon and the first nt of stop codon.
    Determine their 5'-end position (for start codon mappers) and their 3'-end position for
stop codon mappers. Do this seperately for every read length.
    Then identify the offset of the ends to the codons first base for every read length.
This is the p-site offset for start codon mappers and the
    (p-site offset)-3 for the stop codon mappers.
    '''
    count_transcripts = 0
    readlengths = (readlengths[0],readlengths[1]+1)
    shape = (readlengths[1]-2,readlengths[0])
    chloro = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),columns=[x for x in
range(readlengths[0],readlengths[1])],index=pd.Series(data=[x for x in reversed(range(-
shape[0],0))], dtype=int))
    mito = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),columns=[x for x in
range(readlengths[0],readlengths[1])],index=pd.Series(data=[x for x in reversed(range(-
shape[0],0))], dtype=int))
    nuc = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),columns=[x for x in
range(readlengths[0],readlengths[1])],index=pd.Series(data=[x for x in reversed(range(-
shape[0],0))], dtype=int))

    chloro3 = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),columns=[x for x in
range(readlengths[0],readlengths[1])],index=pd.Series(data=[x for x in
reversed(range(1,shape[0]+1))], dtype=int))
    mito3 = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),columns=[x for x in
range(readlengths[0],readlengths[1])],index=pd.Series(data=[x for x in
reversed(range(1,shape[0]+1))], dtype=int))
    nuc3 = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),columns=[x for x in
range(readlengths[0],readlengths[1])],index=pd.Series(data=[x for x in
reversed(range(1,shape[0]+1))], dtype=int))

    for transcript in dataset:
        #print(count_transcripts, " transcripts calculated")
        count_transcripts += 1
        five_utr = dataset[transcript].five_utr
        five_length = sum([abs(x[1]-x[0])+1 for x in five_utr])
        three_utr = dataset[transcript].three_utr
        three_length = sum([abs(x[1]-x[0])+1 for x in five_utr])
        codings = dataset[transcript].codings
        strand = dataset[transcript].strand
        samfile = dataset[transcript].samfile
        chromosome = dataset[transcript].chromosome

        if strand == "+":
            if (five_offset == True) & (five_length >= min_five):
                if chromosome == chloroplast:
                    target = chloro
                elif chromosome == mitochondria:
                    target = mito
                else:
                    target = nuc

                posdict5utr = {}
```

```
                counter = 0
                for i in five_utr:
                    for s in range(i[0],i[1]+1):
                        posdict5utr[s] = counter-five_length
                        counter += 1

                start_nt = min([x for y in codings for x in y])-1 #0-based position of the
first base of start codon

                reads = [[read.pos+1, read.query_alignment_length,read.cigar,
read.reference_end] for read in samfile.fetch(chromosome, start_nt+2, start_nt+3) if
(read.pos < start_nt) & any(read.pos+1 in range(x[0],x[1]+1) for x in five_utr)]
                dfr = pd.DataFrame(reads, columns=["POS","LEN","CIGAR","3END"])

                if dfr["LEN"].count() !=0:
                    dfr["5OFF"] = dfr.apply(lambda row: posdict5utr[row["POS"]] if
row["POS"] in posdict5utr.keys() else "REMOVE", axis=1)
                    for i in dfr[dfr["5OFF"]!="REMOVE"].iterrows():
                        try:
                            target[i[1]["LEN"]].loc[i[1]["5OFF"]]+= 1
                        except KeyError:
                            continue


            if (three_offset == True) & (three_length >= min_three):
                if chromosome == chloroplast:
                    target = chloro3
                elif chromosome == mitochondria:
                    target = mito3
                else:
                    target = nuc3

                posdict3utr = {}
                counter = 1
                for i in three_utr:
                    for s in range(i[0],i[1]+1):
                        posdict3utr[s] = counter
                        counter +=1

                stop_nt = max([x for y in codings for x in y])-3 #0-based position of first
base of stop codon

                reads =
[[read.pos+1,read.query_alignment_length,read.cigar,read.reference_end] for read in
samfile.fetch(chromosome, stop_nt, stop_nt+1) if (read.reference_end-1 > stop_nt) &
any(read.reference_end in range(x[0],x[1]+1) for x in three_utr)]
                dfr = pd.DataFrame(reads, columns=["POS","LEN","CIGAR","3END"])


                if dfr["LEN"].count() !=0:
                    dfr["3OFF"] = dfr.apply(lambda row: posdict3utr[row["3END"]] if
row["3END"] in posdict3utr.keys() else "REMOVE", axis=1)
                    for i in dfr[dfr["3OFF"]!="REMOVE"].iterrows():
                        try:
                            target[i[1]["LEN"]].loc[i[1]["3OFF"]]+= 1
                        except KeyError:
                            continue


        if strand == "-":

            if (five_offset == True) & (five_length >= min_five):
                start_nt = max([x for y in codings for x in y])-1 #0-based position of first
base of start codon
                if chromosome == chloroplast:
                    target = chloro
                elif chromosome == mitochondria:
                    target = mito
                else:
                    target = nuc

                posdict5utr = {}
                counter = 0
                for i in five_utr:
                    for s in reversed(range(i[0],i[1]+1)):
                        posdict5utr[s] = counter-five_length
                        counter += 1
```

```
                reads =
[[read.pos+1,read.query_alignment_length,read.cigar,read.reference_end] for read in
samfile.fetch(chromosome, start_nt-2, start_nt-1) if (read.reference_end-1 > start_nt) &
any(read.reference_end in range(x[0],x[1]+1) for x in five_utr)]
                dfr = pd.DataFrame(reads, columns=["POS","LEN","CIGAR","5END"])

                if dfr["LEN"].count() !=0:

                    dfr["5OFF"] = dfr.apply(lambda row: posdict5utr[row["5END"]] if
row["5END"] in posdict5utr.keys() else "REMOVE", axis=1)
                    for i in dfr[dfr["5OFF"]!="REMOVE"].iterrows():
                        try:
                            target[i[1]["LEN"]].loc[i[1]["5OFF"]]+= 1
                        except KeyError:
                            continue

            if (three_offset == True) & (three_length >= min_three):

                if chromosome == chloroplast:
                    target = chloro3
                elif chromosome == mitochondria:
                    target = mito3
                else:
                    target = nuc3

                stop_nt = min([x for y in codings for x in y])-1 #0-based position of first
base of stop codon

                posdict3utr = {}
                counter = 1
                for i in three_utr:
                    for s in reversed(range(i[0],i[1]+1)):
                        posdict3utr[s] = counter
                        counter += 1

                reads =
[[read.pos+1,read.query_alignment_length,read.cigar,read.reference_end] for read in
samfile.fetch(chromosome, stop_nt+2, stop_nt+3) if (read.reference_end-1 > stop_nt) &
any(read.pos+1 in range(x[0],x[1]+1) for x in three_utr)]
                dfr = pd.DataFrame(reads, columns=["POS","LEN","CIGAR","5END"])

                if dfr["LEN"].count() !=0:
                    dfr["3OFF"] = dfr.apply(lambda row: posdict3utr[row["POS"]] if
row["POS"] in posdict3utr.keys() else "REMOVE", axis=1)
                    for i in dfr[dfr["3OFF"]!="REMOVE"].iterrows():
                        try:
                            target[i[1]["LEN"]].loc[i[1]["3OFF"]]+= 1
                        except KeyError:
                            continue


    return [chloro, mito, nuc, chloro3, mito3, nuc3]

def FPlength_dist(sample, genomic_coordinates, readlengths=(20,39),
organelles=["plastome","mitogenome","chrM","mito","CP","Cp","Mt","MT","chloroplast","mitocho
ndrium","plastid","mitochondria","plast"]):

    chromosomes = list(genomic_coordinates.chromosome.unique())
    shape=(readlengths[1]-readlengths[0]+1, len(chromosomes))

    counts = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),index=[x for x in
range(readlengths[0],readlengths[1]+1)],columns=chromosomes)
    sample = ps.AlignmentFile(sample)

    for s in chromosomes:
        x = [i.query_length for i in sample.fetch(s)]
        x = [i for i in x if ((i<=readlengths[1]) & (i>=readlengths[0]))]
        y = np.unique(np.array(x), return_counts=True)
        for ind, count in zip(y[0],y[1]):
            counts.loc[ind,s] = count
    counts["cytosolic"]=counts.apply(lambda row: row.drop(columns=organelles,
errors="ignore").sum(), axis=1)
    normalized = counts.apply(lambda col: col/col.sum(), axis=0)
    return [counts, normalized]

def multiplot2(identifier, datasets, how="together",path=False, names=False, label="A-B",
mode="difference", on="cds"): #just annotate the identifier, the savepath and the list of
datasets to comapre, currently only possible in cds mode
```

```
    base = datasets[0]
    transcript = base[identifier]
    identifier = transcript.identifier
    name = transcript.name
    if name == "None":
        name = identifier
    if path != False:
        path = path+name+".pdf"
    newcov = []
    plt.clf()
    plt.style.use("seaborn-talk")
    if on == "cds":
        for data in datasets:
            try:
                cov = data[identifier].cds_coverage
                reads = data[identifier].cds_reads
                total = sum(data[identifier].cds_coverage.values())

            except:
                data[identifier].only_cds()
                #data[identifier].export_profile()
                cov = data[identifier].cds_coverage
                reads = data[identifier].cds_reads
                total = sum(data[identifier].cds_coverage.values())
            if total == 0:
                ncov=list(len(cov)*[0])
            else:
                ncov=list(map(lambda x: x/total*100,cov.values()))
            newcov.append(ncov)
    elif on=="full":
        for data in datasets:
            try:
                cov = data[identifier].full_coverage
                reads = data[identifier].full_reads
                total = sum(data[identifier].full_coverage.values())

            except:
                data[identifier].fulltranscript()
                #data[identifier].export_profile()
                cov = data[identifier].full_coverage
                reads = data[identifier].full_reads
                total = sum(data[identifier].full_coverage.values())
            if total == 0:
                ncov=list(len(cov)*[0])
            else:
                ncov=list(map(lambda x: x/total*100,cov.values()))
            newcov.append(ncov)
    fig = plt.figure()
    fig.set_figheight(5)
    fig.set_figwidth(10)
    ax = fig.add_subplot(1,1,1)
    if how=="together":
        if names == False:
            names = list(range(len(datasets)))
            print(names)
        #for i,n in zip(newcov,names):
        #    ax.plot(i, label=n)
        for i, count in zip(datasets,range(0,len(datasets))):
            ax.plot(newcov[count],label=names[count])
        #ax.plot(newcov[1],label=names[1], color="#1971ffff")
        ax.legend()
        ax.set_ylabel("normalized coverage [%]")
        if on=="cds":
            ax.set_xlabel("position from start codon [nt]")
        elif on == "full":
            ax.set_xlabel("position from TSS [nt]")
    elif how=="versus":
        if mode == "difference":
            plotty = [i-j for i,j in zip(newcov[0],newcov[1])]
            liney = 0
            ylabel = "difference"
        elif mode == "ratio":
            plotty = [i/j if j > 0 else i/sum(newcov[0][:90])/90 if ((sum(newcov[0][:90])/90
> 0)&(newcov[0][:90].count(0)<= 45))else i/(base[identifier].cds_rpkm/1000) if
base[identifier].cds_rpkm >0 else "E" for i,j in zip(newcov[0],newcov[1])]
            liney = 1
            ylabel = "ratio"
        if "E" in plotty:
```

```
            print("Zero-division error in ",identifier)
            return
        ax.plot(plotty, label=label)
        if mode == "ratio":
            ax.set_yscale("log", base=2)
        #ax.spines["bottom"].set_position(("data",0))
        ax.axhline(y=liney, color="red")
        ax.spines["right"].set_color("none")
        ax.spines["top"].set_color("none")
        ax.legend()
        if mode == "ratio":
            ax.set_ylim(0.3)
        plt.ylabel(ylabel)
        plt.xlabel("position from start codon [nt]")
    figuretitle = str(transcript.name)+"\n("+str(transcript.identifier)+")"
    identifier = str(transcript.identifier)
    if name != "None":
        plt.title(label =figuretitle)
    else:
        plt.title(label = identifier)
    if path != False:
        fig.savefig(path, format="pdf")
    plt.show()
    return

def fasta_to_dict(fasta, headersymbol=">"): #With this function you can convert a common
FASTA genome assembly into a dictionary with chromosome headers as keys and sequences as
headers
    genome = {}
    seqlines = []
    count = 0
    with open(fasta, "r") as lines:
        for line in lines.readlines():
            if count == 0:
                if line[0] == headersymbol:
                    name = line[1::].strip("\n")
                    print(name)
                    count += 1
            else:
                if line[0] == headersymbol:
                    x = ""
                    for n in seqlines:
                        x = x+n
                    seqlines = []
                    genome[name] = x
                    name = line[1::].strip("\n")
                else:
                    seqlines.append(line.strip("\n"))

    return genome

def grab(genome, identifier, dataset, spec = False, start=0, stop=0, feature="coding"):
    transcript = dataset[identifier]

    chrom = transcript.chromosome
    strand = transcript.strand

    five = transcript.five_utr
    three = transcript.three_utr

    if feature == "coding":
        codings = transcript.codings
    elif feature == "all":
        codings = transcript.five_utr+transcript.codings+transcript.three_utr
    elif feature == "five_utr":
        codings = transcript.five_utr
    elif feature == "three_utr":
        codings = transcript.three_utr
    else:
        print("Please define valid genomic feature: \"coding\", \"five_utr\", \"three_utr\"
or \"all\". \"coding\" is the predefined feature to extract.")
        return
    coords = []
    if len(codings) > 1:
        codingseq = ""
        for x in sorted(codings):
            codingseq += genome[chrom][x[0]-1:x[1]]
            coords+= list(range(x[0],x[1]+1))
```

```
    elif len(codings)==1:
        codingseq = genome[chrom][codings[0][0]-1:codings[0][1]]
        coords+= list(range(codings[0][0],codings[0][1]+1))
    elif len(codings)==0:
        return "", "", "", "", "", (start, stop), ""


    if strand == "-":
        codingseq = codingseq[::-1]
        codingseq = codingseq.replace("A","X")
        codingseq = codingseq.replace("T","A")
        codingseq = codingseq.replace("X","T")
        codingseq = codingseq.replace("G","Y")
        codingseq = codingseq.replace("C","G")
        codingseq = codingseq.replace("Y","C")

    fullcds = codingseq
    if spec == False:
        originalseq = codingseq
    elif spec == True:
        originalseq = codingseq[start-1:stop]
        start = start - 1
        remainder = start % 3
        if remainder != 0:
            start = start-remainder
        remainder = stop % 3
        if remainder != 0:
            stop = stop+(3-remainder)

        codingseq = codingseq[start:stop]
    rnaseq = codingseq.replace("T","U")

    translation =
{"TTT":"F","TTC":"F","TTA":"L","TTG":"L","CTT":"L","CTC":"L","CTA":"L","CTG":"L","ATT":"I","
ATC":"I","ATA":"I","ATG":"M","GTT":"V","GTC":"V","GTA":"V","GTG":"V","TCT":"S",

"TCC":"S","TCA":"S","TCG":"S","CCT":"P","CCC":"P","CCA":"P","CCG":"P","ACT":"T","ACC":"T","A
CA":"T","ACG":"T","GCT":"A","GCC":"A","GCA":"A","GCG":"A","TAT":"Y","TAC":"Y",

"TAA":"*","TAG":"*","CAT":"H","CAC":"H","CAA":"Q","CAG":"Q","AAT":"N","AAC":"N","AAA":"K","A
AG":"K","GAT":"D","GAC":"D","GAA":"E","GAG":"E","TGT":"C","TGC":"C","TGA":"*",

"TGG":"W","CGT":"R","CGC":"R","CGA":"R","CGG":"R","AGT":"S","AGC":"S","AGA":"R","AGG":"R","G
GT":"G","GGC":"G","GGA":"G","GGG":"G"}
    if feature=="coding":
        aaseq = "".join([translation[codon] for codon in [codingseq[i:i+3] for i in
range(0,len(codingseq),3)]])
    else:
        aaseq="extracted feature was non-coding."
    return [codingseq, aaseq, rnaseq, originalseq, fullcds, (start, stop), coords]

def peakfinder(peaklist):
    peaks = []
    detect = False
    idxlist = list(range(1,len(peaklist)))
    if len(idxlist) == 0:
        maxidx = 0
    else:
        maxidx = max(idxlist)

    for x,idx in zip(peaklist[1::],idxlist):
        if detect == True:
            if x-peaklist[idx-1] != 1:
                detect = False
                peaks.append((peakstart, peaklist[idx-1]))
            elif ((idx == maxidx)&(x-peaklist[idx-1] == 1)):
                detect = False
                peaks.append((peakstart, x))
            else:
                continue
        elif detect == False:
            if x-peaklist[idx-1] == 1:
                peakstart = peaklist[idx-1]
                detect = True
    return peaks

def compare_profiles(identifier, controls, treated, normalization="scale", scaleto=
10000000, on="cds"):
```

```
    ctrl_normalized = {}
    treated_normalized = {}

    for group in [controls, treated]:
        counter = 1
        for data in group:
            scalefactor = scaleto / data[identifier].totreads
            if on == "cds":
                try:
                    cov = data[identifier].cds_coverage
                    reads = data[identifier].cds_reads
                    total = sum(data[identifier].cds_coverage.values())

                except:
                    data[identifier].only_cds()
                    #data[identifier].export_profile()
                    cov = data[identifier].cds_coverage
                    reads = data[identifier].cds_reads
                    total = sum(data[identifier].cds_coverage.values())

            elif on == "full":
                try:
                    cov = data[identifier].full_coverage
                    reads = data[identifier].full_reads
                    total = sum(data[identifier].full_coverage.values())

                except:
                    data[identifier].fulltranscript()
                    #data[identifier].export_profile()
                    cov = data[identifier].full_coverage
                    reads = data[identifier].full_reads
                    total = sum(data[identifier].full_coverage.values())

            if normalization == "internal":
                if total == 0:
                    ncov=list(len(cov)*[0])
                else:
                    ncov=list(map(lambda x: x/total*100,cov.values()))

            elif normalization == "scale":
                ncov=list(map(lambda x: x*scalefactor,cov.values()))

            if data in controls:

                ctrl_normalized[str(counter)] = ncov
                counter += 1
            elif data in treated:

                treated_normalized[str(counter)] = ncov
                counter += 1

    control_df = pd.DataFrame(data = ctrl_normalized)
    treated_df = pd.DataFrame(data = treated_normalized)

    control_df["mean"] = control_df.apply(lambda row: row.mean(axis=0), axis=1)
    control_df["SD"] = control_df.apply(lambda row: row.drop(["mean"]).std(axis=0), axis=1)

    treated_df["mean"] = treated_df.apply(lambda row: row.mean(axis=0), axis=1)
    treated_df["SD"] = treated_df.apply(lambda row: row.drop(["mean"]).std(axis=0), axis=1)

    return [treated_df, control_df]

def biotype_length_dist(sample, genomic_coordinates, readlengths=(20,39),
organelles=["plastome","mitogenome","chrM","mito","CP","Cp","Mt","MT","chloroplast","mitocho
ndrium","plastid","mitochondria","plast"], threshold=0.5, chromosomes=False, binsize=10000,
fast=True, center=True):
    if chromosomes == False:
        chromosomes = list(genomic_coordinates.chromosome.unique())
    shape=(readlengths[1]-readlengths[0]+1, len(chromosomes))
    print(len(chromosomes), " contigs to process.")

    def detect_match(pos,cigar):
        currpos = pos
        matching_regions = []
        for tup in cigar:
            if tup[0] in [0,7,8]:
                for pos in range(currpos,currpos+tup[1]):
                    matching_regions.append(pos)
```

```
            elif tup[0] == [1,4,5,6]:
                continue
            else:
                currpos+=tup[1]
        return matching_regions

    coding_result = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),index=[x for x in
range(readlengths[0],readlengths[1]+1)],columns=chromosomes)
    five_result = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),index=[x for x in
range(readlengths[0],readlengths[1]+1)],columns=chromosomes)
    three_result = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),index=[x for x in
range(readlengths[0],readlengths[1]+1)],columns=chromosomes)
    noncoding_result = pd.DataFrame(data= np.zeros(shape=shape, dtype=int),index=[x for x in
range(readlengths[0],readlengths[1]+1)],columns=chromosomes)

    sample = ps.AlignmentFile(sample)
    number = len(chromosomes)
    chr_count = 1
    genome = genomic_coordinates
    for s in chromosomes:

        reads = {}
        r_count = 0
        for x in sample.fetch(s):
            reads[r_count] = [x.seq, len(x.seq), x.pos, x.cigar]
            r_count += 1
        if len(reads) == 0:
            continue
        reads_df = pd.DataFrame.from_dict(reads, orient="index",
columns=["seq","len","pos","cigar"])

        if fast == True:
            if center == True:
                reads_df["matching"] = reads_df.apply(lambda row:
detect_match(row["pos"],row["cigar"]), axis=1)
                reads_df["pos"] = reads_df.apply(lambda row:
row["matching"][int(len(row["matching"])/2)], axis=1)
            read_max = reads_df["pos"].max()
        elif fast == False:
            reads_df["matching"] = reads_df.apply(lambda row:
detect_match(row["pos"],row["cigar"]), axis=1)
            reads_df["max"] = reads_df.apply(lambda row: max(row["matching"]), axis=1)
            read_max = reads_df["max"].max()


        print("starting "+str(s)+". "+str(chr_count)+"/"+str(number))
        chr_count += 1
        codings = genome[genome["chromosome"]==s]["codings"].tolist()
        codings = list(set(it.chain.from_iterable(codings)))
        codings =
np.unique(np.array(list(it.chain.from_iterable([list(range(tup[0],tup[1]+1)) for tup in
codings]))))
        fives = genome[genome["chromosome"]==s]["five_prime_UTR"].tolist()
        fives = list(set(it.chain.from_iterable(fives)))
        fives = np.unique(np.array(list(it.chain.from_iterable([list(range(tup[0],tup[1]+1))
for tup in fives]))))
        threes = genome[genome["chromosome"]==s]["three_prime_UTR"].tolist()
        threes = list(set(it.chain.from_iterable(threes)))
        threes =
np.unique(np.array(list(it.chain.from_iterable([list(range(tup[0],tup[1]+1)) for tup in
threes]))))

        cds_max = max(codings) if len(codings)>0 else 1
        five_max = max(fives) if len(fives)>0 else 1
        three_max = max(threes) if len(threes)>0 else 1
        maximum = max(cds_max, five_max, three_max ,read_max)


        cds_bins = {}
        cds_count = 1
        for x in range(math.ceil(maximum/binsize)):
            cds_bins[cds_count] = codings[(codings<= (cds_count*binsize))&(codings >
(cds_count-1)*binsize)]
            cds_count +=1


        five_bins = {}
        f_count = 1
```

```
        for x in range(math.ceil(maximum/binsize)):
            five_bins[f_count] = fives[(fives<= (f_count*binsize))&(fives > (f_count-
1)*binsize)]
            f_count +=1


        three_bins = {}
        t_count = 1
        for x in range(math.ceil(maximum/binsize)):
            three_bins[t_count] = threes[(threes<= (t_count*binsize))&(threes > (t_count-
1)*binsize)]
            t_count +=1

        if fast == True:
            reads_df["bin"] = reads_df.apply(lambda row: math.ceil(row["pos"]/binsize),
axis=1)
            reads_df["cds_matched"] =reads_df.apply(lambda row: True if np.any(row["pos"] ==
cds_bins[row["bin"]]) else False, axis=1)
            reads_df["five_matched"] =reads_df.apply(lambda row: False if row["cds_matched"]
== True else True if np.any(row["pos"] == five_bins[row["bin"]]) else False, axis=1)
            reads_df["three_matched"] =reads_df.apply(lambda row: False if
((row["cds_matched"] == True) | (row["cds_matched"] == True)) else True if np.any(row["pos"]
== three_bins[row["bin"]]) else False, axis=1)
            reads_df["intergenic_matched"] = reads_df.apply(lambda row: True if
((row["cds_matched"]==False)&(row["five_matched"]==False)&(row["three_matched"]==False))
else False, axis=1)
            #reads_df["intergenic_matched"] = reads_df.apply(lambda row: False if
any(row[["cds_matched","five_matched","three_matched"]]==True) else True, axis=1)

            coding_reads = np.unique(reads_df[reads_df["cds_matched"]==True]["len"],
return_counts=True)
            five_utr_reads = np.unique(reads_df[reads_df["five_matched"]==True]["len"],
return_counts=True)
            three_utr_reads = np.unique(reads_df[reads_df["three_matched"]==True]["len"],
return_counts=True)
            intergenic_reads = np.unique(reads_df[reads_df["intergenic_matched"]==
True]["len"], return_counts=True)

        else:

            reads_df["bin"] = reads_df.apply(lambda row: list(set([math.ceil(x/binsize) for
x in row["matching"]])), axis=1)

            reads_df["cds_matched"] =reads_df.apply(lambda row: len([x for x in
row["matching"] if any(x in cds_bins[b] for b in row["bin"])])/len(row["matching"]), axis=1)
            reads_df["five_matched"] =reads_df.apply(lambda row: len([x for x in
row["matching"] if any(x in five_bins[b] for b in row["bin"])])/len(row["matching"]),
axis=1)
            reads_df["three_matched"] =reads_df.apply(lambda row: len([x for x in
row["matching"] if any(x in three_bins[b] for b in row["bin"])])/len(row["matching"]),
axis=1)
            reads_df["intergenic_matched"] = 1-
(reads_df["cds_matched"]+reads_df["five_matched"]+reads_df["three_matched"])

            coding_reads = np.unique(reads_df[reads_df["cds_matched"]>= threshold]["len"],
return_counts=True)
            five_utr_reads = np.unique(reads_df[reads_df["five_matched"]>=
threshold]["len"], return_counts=True)
            three_utr_reads = np.unique(reads_df[reads_df["three_matched"]>=
threshold]["len"], return_counts=True)
            intergenic_reads = np.unique(reads_df[reads_df["intergenic_matched"]>=
threshold]["len"], return_counts=True)


        for x, biotype in zip([coding_reads, five_utr_reads, three_utr_reads,
intergenic_reads],[coding_result, five_result, three_result, noncoding_result]):
            for ind, count in zip(x[0],x[1]):
                biotype.loc[ind,s] = count
    for biotype in [coding_result, five_result, three_result, noncoding_result]:
        biotype["cytosolic"]=biotype.apply(lambda row: row.drop(columns=organelles,
errors="ignore").sum(), axis=1)
    sample.close()
    return [coding_result, five_result, three_result, noncoding_result]
```

8.4 NGS raw data processing pipeline

```
#!/bin/bash
'
This pipeline is used to process raw fastq files from a Ribo-Seq Illumina sequencing run and
map the reads against the Chlamy genome.
The pipeline works by using different environments in conda that contain different tools and
a custom python script.

The single steps in this pipeline are:
1. conda is initialized and a folder containing all raw files to process is defined.
2. The script iteratively calls programs in sequential order for each raw file in the
defined folder.
       2.1. Cutadapt is called to remove adapter sequences from the 3-prime of each read.
       2.2. umi_tools is called to remove the first 4 nt 5prime of each read and adds them
to the read identifier as unique molecular identifier (UMI).
       2.3. umi_tools is called a second time, this time to remove the last 4 nt 3prime of
each read and also adds them to the already tagged UMI (like this: IDENTIFIER_UMI5_UMI3).
       2.4. cutadapt is called a second time, this time to remove all reads out of the
range 20-39 nt. This is the earliest time point this step makes sense. Could also be done at
a later step, but my intention was to reduce the amount of data as early as possible for
better efficiency.
       2.5. my custom python script umi_correction.py is called which simply iterates over
every read in the file and modifies the read identifiers from IDENTIFIER_UMI5_UMI3 to
IDENTIFIER_UMI5UMI3. This is necessary for the deduplication step later since umi_tools
interprets the underscore as delimiter between read identifier and UMI, which leads to
falsely identifying only UMI3 as UMI if the underscore between both is not removed.
       2.6. STAR is called and maps the remaining reads against the ncRNA sequences. This
is done in two-pass mode without annotation for simplicity. Reads that do not map to the
ncRNAs are output to a new file.
       2.7. STAR is called a second time, this time taking all reads that did not map to
the ncRNAs and maps them against the Chlamy genome with annotation. Unmapped reads are
output again, although not further used in this pipeline (at the moment).
       2.8. samtools is called to index the resulting bam file from STAR. The index is
needed for umi_tools in the next step as well as downstream analysis with plastid.
       2.9. umi_tools is called to browse the bam file for read duplicates (all reads
having the same sequence + the same UMI tags in the header) and removes all duplicates
except for one read.
       2.10. done. The iteration is finished and will start again from the beginning with
the next file in the defined folder until all files have been processed.
3. All finished. The script prints out "ALL DONE" to the console and exits.
'
#1 - initialize conda, define adapter sequence to clip and define folder containing the raw
reads

. /software/anaconda3/latest/etc/profile.d/conda.sh
while true; do
read -p "Do you whish to perform mapping against ncRNA decoys? (Type y/n and enter)" yn
case $yn in
[yY] ) ncRNAfilter="true";break;;
[nN] ) ncRNAfilter="false"; ncRNAgenome="No genomic information";break;;
esac
done

if [[ $ncRNAfilter == "true" ]]
then
       ncRNAgenome=$(zenity --file-selection --title="Select a folder containing the STAR
genome for ncRNA removal" --directory)
fi

echo -e "\e[92m $ncRNAgenome selected for ncRNA mapping.\e[0m"
mappinggenome=$(zenity --file-selection --title="Select a folder containing the STAR genome
for mapping" --directory)
echo -e "\e[92m $mappinggenome selected for genome mapping.\e[0m"
input=$(zenity --file-selection --title="Select a folder containing gzipped FASTQ files for
processing" --directory)
echo -e "\e[92m $input selected as location of raw input files.\e[0m"
files=${input}/*
folder=$(zenity --file-selection --title="Select or create a folder to save the processed
data in." --directory)
echo -e "\e[92m $folder as output folder.\e[0m"


trimmedf=${folder}/trimmed/
```

```
mkdir $trimmedf

umi5f=${folder}/umi5/
mkdir $umi5f

umi53f=${folder}/umi53/
mkdir $umi53f

sizefilterf=${folder}/sizefilter/
mkdir $sizefilterf

ncRNAmappedf=${folder}/ncRNAmapped/
mkdir $ncRNAmappedf

genome_mappedf=${folder}/genome_mapped/
mkdir $genome_mappedf

deduplicatedf=${folder}/deduplicated/
mkdir $deduplicatedf

#2 - iterate over raw read files

for f in $files
do

#2.1 - activate cutadapt and clip the 3'-adapter sequence, also save the name of the sample
to variable SAMPLE
conda activate cutadapt

name="$(basename -- $f)"
sample="${name%.fastq.gz}"
echo "sample name: ${sample}"
current_file=$f

out=$trimmedf
adapter_trimmed="${out}${sample}_adapter_trimmed.fastq.gz"
trimlog="${out}${sample}_log.txt"
adapter_sequence="TGGAATTCTCGGGTGCCAAGG"

echo -e "\e[92mstart adapter trimming!\e[0m"
echo "current file: $current_file"
echo "output directory: $adapter_trimmed"
(cutadapt --cores=20 --minimum-length 9 -a "$adapter_sequence" -o "$adapter_trimmed"
"$current_file") 1> "$trimlog"
echo -e "\e[92mAdapters removed!\e[0m"

#2.2 and 2.3 - activate umi_tools and extract 5'-UMI in a first run, then pass the resulting
file to a second run extracting the 3'-UMI

conda activate umi_tools

echo -e "\e[92mStart 5'-UMI extraction.\e[0m"
out=$umi5f
umi5="${out}${sample}_umi5.fastq.gz"
umilog="${out}${sample}_umi5.log"

umi_tools extract --stdin="$adapter_trimmed" --bc-pattern=NNNN --log="$umilog" --stdout
"$umi5"

echo -e "\e[92mStart 3'-UMI extraction.\e[0m"
out=$umi53f
umi53="${out}${sample}_umi53.fastq.gz"
umilog="${out}${sample}_umi53.log"

umi_tools extract --stdin="$umi5" --bc-pattern=NNNN --3prime --log="$umilog" --
stdout="$umi53"

echo -e "\e[92m UMIs extracted\e[0m"

#2.4 - again activate cutadapt and remove all reads out of the size range 20-39 nt

conda activate cutadapt

echo -e "\e[92m Start size filtering.\e[0m"

out=$sizefilterf
sizefiltered="${out}${sample}_sizefiltered.fastq.gz"
sizelog="${out}${sample}_sizefilter_log.txt"
```

```
(cutadapt --cores=20 --minimum-length 20 --maximum-length 39 -o "$sizefiltered" "$umi53") 1>
"$sizelog"
echo -e "\e[92m reads filtered by length.\e[0m"

#2.5 - activate Jupyter and run the custom python script to correct the UMI tags in the
remaining reads

conda activate Jupyter

echo -e "\e[92m UMI-tag correction.\e[0m"
python /home/gotsmann/Desktop/final\ pipelines/umi_correction.py "$sizefiltered"
echo -e "\e[92m UMI tags corrected.\e[0m"

#2.6 and 2.7 - activate STAR and perform first the ncRNA mapping, then rename the file with
unmapped reads (to keep the file name a bit simpler) and take it as input for mapping agains
the full genome

conda activate star

if [[ "$ncRNAfilter" == "true" ]]
then
        corrected="${out}${sample}_sizefiltered_corrected.fastq.gz"
        output=$ncRNAmappedf
        ncRNAmapped="${output}${sample}_ncRNAmapped_"

        STAR --runMode alignReads --runThreadN 16 --genomeDir "$ncRNAgenome" --readFilesIn
"$corrected" --outFileNamePrefix "$ncRNAmapped" --outSAMtype BAM SortedByCoordinate --
outFilterMultimapNmax 20 --outFilterMismatchNmax 3 --outReadsUnmapped Fastx --alignIntronMax
7000 --limitBAMsortRAM 32000000000 --readFilesCommand zcat --twopassMode Basic

        unmapped="${ncRNAmapped}Unmapped.out.mate1"
        output=$genome_mappedf
        genomemapped="${output}${sample}_genome_mapped_"

        STAR --runMode alignReads --runThreadN 16 --genomeDir "$mappinggenome" --readFilesIn
"$unmapped" --outFileNamePrefix "$genomemapped" --outReadsUnmapped Fastx --
outFilterMismatchNoverLmax 0.1 --alignIntronMax 8000 --outSAMmultNmax 1 --
outMultimapperOrder Random --outSAMtype BAM SortedByCoordinate --limitBAMsortRAM 32000000000
        echo -e "\e[92m genome mapping done.\e[0m"
        #OUTPUT=/scratch/gotsmann/Seq_Data/footprints/deduplication/aligned/
        aligned="${genomemapped}Aligned.sortedByCoord.out.bam"
        #NEWNAME="${OUTPUT}${SAMPLE}_ALIGNED.bam"
        #mv $UNMAPPED $NEWNAME

else
        echo "ncRNA mapping skipped due to user specification."
        unmapped="${out}${sample}_sizefiltered_corrected.fastq.gz"
        output=$genome_mappedf
        genomemapped="${output}${sample}_genome_mapped_"

        STAR --runMode alignReads --runThreadN 16 --genomeDir "$mappinggenome" --readFilesIn
"$unmapped" --outFileNamePrefix "$genomemapped" --outReadsUnmapped Fastx --
outFilterMismatchNoverLmax 0.1 --alignIntronMax 8000 --outSAMmultNmax 1 --
outMultimapperOrder Random --outSAMtype BAM SortedByCoordinate --limitBAMsortRAM 32000000000
--readFilesCommand zcat
        echo -e "\e[92m genome mapping done.\e[0m"
        #OUTPUT=/scratch/gotsmann/Seq_Data/footprints/deduplication/aligned/
        aligned="${genomemapped}Aligned.sortedByCoord.out.bam"
        #NEWNAME="${OUTPUT}${SAMPLE}_ALIGNED.bam"
        #mv $UNMAPPED $NEWNAME
fi

#2.8 - samtools indexing

conda activate Jupyter
echo -e "\e[92m SAMtools indexing.\e[0m"
samtools index "$aligned"
echo -e "\e[92m bam file indexed.\e[0m"
#2.9 - umi_tools deduplication

conda activate umi_tools
output=$deduplicatedf
log="${output}${sample}_log.txt"
outfile="${output}${sample}_deduplicated.bam"
echo -e "\e[92m Start deduplication.\e[0m"
umi_tools dedup --stdin="$aligned" --log="$log" > "$outfile"
echo -e "\e[92m bam file deduplicated.\e[0m"
```

```
#2.10 - samtools indexing of deduplicated data sets

conda activate Jupyter
echo -e "\e[92m SAMtools indexing.\e[0m"
samtools index "$outfile"
echo -e "\e[92m bam file indexed.\e[0m"

done
echo -e "\e[92mALL DONE! \e[0m"
```

8.5 Python script for correction of read headers

```
import gzip
import sys
path=sys.argv[1]
name=(path.split("/")[-1].split(".fastq.gz")[0])+"_corrected.fastq.gz"
parts= path.rsplit("/",1)
out=parts[0]+"/"+name
with gzip.open(path,"rt") as reads:
    with gzip.open(out, "wb") as corrected:
        for line in reads:
            if line[0]=="@":
                x = line.split("_")
                umi = x[1]+x[2]
                line = x[0]+"_"+umi
            corrected.write(line.encode())
```

8.6 List of main figures

8.7 List of main tables

8.8 List of abbreviations

°C	Degree Celsius
µ	Micro, unit prefix
A	Ampere
A	Average (in context of MA plots)
A. thaliana	*Arabidopsis thaliana*
A/T/G/C/U	Nucleosides (adenosine/thymidine/guanosine/cytidine/uridine)
APS	Ammonium persulfate
aSD	anit-Shine-Dalgarno
A-site	Aminoacyl-site
ATP	Adenosine triphosphate
ATPase	ATP synthase
BAM	Binary alignment map
BR	Borad range
BSA	Bovine serum albumine
Btn	Biotin
c	Centi, unit prefix
C. r.	*Chlamydomonas reinhardtii*
C. reinhardtii	*Chlamydomonas reinhardtii*
C2 acids	Organic acids containing two carbon atoms
C4 acid	Organic acids containing four carbon atoms
Ca	Calcium
CAP	Chloramphenicol
CCM	Carbon concentrating mechanism
CDS	Coding sequence
CES	Control of epistasy by synthesis
CH_3COOK	Potassium acetate
CHX	Cycloheximide
Cl	Chlorine
cm	Centimeter
CO_2	Carbon dioxide
Cp	Chloroplast
CPM	Counts per million
CPU	Computing processing unit
Cre	Chlamydomonas reinhardtii (in context of genomic annotation)
C-terminus	Carboxy-terminus
ctrl	Control sample
Cu	Copper
Cyt	Cytosol
D	Dalton (in context of mass)
DE	Differential expression
DEPC	Dimethyl pyrocarbonate
DMP	Dimethyl pimelimidate
DNA	Desoxyribonucleic acid
dsDNA	double-stranded DANN (see DNA)
DSP	Dithiobissuccinimidyl propionate
DTT	Dithiothreitol
E	Einstein
E. c.	*Escherichia coli*
E. coli	*Escherichia coli*
e.g.	example given
ECL	Enhanced chemiluminescence
EDTA	Ethyldiamin tetraacetic acid
E-site	Exit-site
ETC	Electron transport chain
f	Femto, unit prefix

FASTA	FAST-All (sequence format)
FASTQ	FASTA-Quality (sequence format)
FC	Fold change
FDR	False discovery rate
Fe	Iron
FW	Felix Willmund (in context of oligo naming)
g	Gram
g	9.81 m/s^2
GB	Gigabyte
GFF3	General feature format 3
GO	Gene ontology
GTP	Guanosine triphosphate
h	Hours
H	Hydrogen
H^+	Proton
H_2O	Water
HEPES	2-[4—(2-Hydroxyethyl)piperazin-1-yl] ethylsulfonic acid
HMP	HEPES-minimal-phpsphate medium
HS	High sensitivity
Hz	Hertz
IC	Initiation complex
k	Kilo, unit prefix
K	Potassium
KCl	Potassium chloride
L	Litre
L	Large subunit (in context of ribosomal protein naming)
LHC	Light harvesting complex
LHCPs	Light harvesting complex proteins
$\log_x$	Logarithmic transformation to the base of x
LOWESS	Locally weighted scatterplot smoothing
LSU	Large (ribosomal) subunit
LUCA	Last universal common ancestor
M	Molar (in context of concentrations)
m	Milli, unit prefix
M	Millions (in context of counting)
M	Minus (in context of MA plots)
m^6A	6'-Methyl adenosine
mfe	Minimum free energy
Mg	Magnesium
$Mg(CH_3COO)_2$	Magnesium acetate
$MgCl_2$	Magnesium Dichloride
miRNA	micro RNA (see RNA)
Mn	Manganese
MNase	Micrococcal nuclease
Mo	Molybdenium
mRNA	messenger RNA (see RNA)
MS	Mass spectrometry
Mt	Mitochondrion
n	Nano, unit prefix
N	Nitrogen
Na	Sodium
NaCl	Sodium chloride
NADH	Nicotinamide adenine-dinucleotide hydrid
NADP	Nicotinamide adenine-dinucleotide phosphate
NaOH	Sodium hydroxide
ncRNA	non-coding RNA (see RNA)
NGS	Next Generation Sequencing

NP-40	Nonidet P-40 (detergent)
nt	Nucleotide
N-terminus	Amino-terminus
Nuc	Nucleus
O	Oxygen
OD_x	Optical density at wavelength x nm
Oligo	Oligonucleotide
ORF	Open reading frame
p	Pico, unit prefix
PAGE	Polyacrylamide gel electrophoresis
PBS	Phosphate buffered saline
PCI	Phenol-chloroform isoamylalcohol
PCR	Polymerase chain reaction
pcSILAC	pulse chase SILAC (see SILAC)
pe	Polypeptide tunnel exit
pH	Pondus hydrogenii (index for measuring acidity of solutions)
PIC	Preinitiation complex
PNK	Polynucleotide kinase
PS	Photosystem
PS I	Photosystem 1
PS II	Photosystem 2
P-site	Peptidyl-site
PSRPs	Plastid-specific ribosomal proteins
p-value	Probability value
QC	Quality control
Ribo-Seq	Ribosome Sequencing
RMA	Random access memory
RNA	Ribonucleic acid
RNAi	RNA interference
RNAP	RNA-Polymerase
RNase	Ribonuclease
RNA-Seq	RNA-Sequencing
ROS	Reactive oxygen species
RPF	Ribosome protected fragment
RPKM	Reads per kilobase per million
r-protein	Ribosomal protein
rRNA	ribosomal RNA (see RNA)
Rsf	Ribosome functions
RSME	Root square mean error
s	Second
S	Svedberg
S	Sulfur
S	Small subunit (in context of ribosomal protein naming)
SAM	Simple alignment map
SD	Shine-Dalgarno
SDS	Sodium dodecylsulfate
Se	Selenium
seRP	Selective ribosome profiling
SILAC	Stable isotope labeling by amino acids
siRNA	small interfering RNA (see RNA)
sRNA	small RNA (see RNA)
SRP	Signal recognition particle
SSC-buffer	Sodium chloride/sodium citrate buffer
SSU	Small (ribosomal) subunit
TAP	Tris-acetate-phosphate medium
TBE	Tris-Borate-EDTA
TEMED	Tetramethylethylen diamine

TIC	Chloroplast inner envelope translocon
TKM	Tris-Potassium-Magnesium
TKM-D	TKM-Dithiothreitol
TKM-T	TKM-Tween
TMD	Transmembrane domain
TMP	Tris-minimal-phosphate medium
TOC	Chloroplast outer envelope translocon
Tris-HCL	Hydrochloric acid buffered Tris
TSS	Transcription start site
tRNA	transfer RNA (see RNA)
U	Unit
UMI	Unique molecular identifier
UTR	Untranslated region
V	Volt
w/v	Weight per volume
XR	Extended range
Zn	Zinc

8.9 List of publications

The following list contains all scientific publications that I contributed to during my doctorate (2018 – 2025) in decreasing chronological order.

Ting, M.K.Y., Gao, Y., Barahimipour, R., Ghandour, R., Liu, J., Martinez-Seidel, F., Smirnova, J., **Gotsmann, V.L.**, Fischer, A., Haydon, M.J., Willmund, F., Zoschke, R., (2024), Optimization of ribosome profiling in plants including structural analysis of rRNA fragments. Plant Methods 20, 143. https://doi.org/10.1186/s13007-024-01267-3

Gotsmann, V.L., Ting, M.K.Y., Haase, N., Rudorf, S., Zoschke, R., Willmund, F., (2024), Utilizing high-resolution ribosome profiling for the global investigation of gene expression in *Chlamydomonas*. The Plantz Journal 117, pp. 1614-1634. https://doi.org/10.1111/tpj.16577

Chunduri, N.K., Menges, P., Zhang, X., Wieland, A., **Gotsmann, V.L.**, Mardin, B.R., Buccitelli, C., Korbel, J.O., Willmund, F., Kschischko, M., Räschle M., Storchová, Z., (2021), Systems approaches identify the consequences of monosomy in somatic human cells. Nature Communications 12, 5576. https://doi.org/10.1038/s41467-021-25288-x

Westrich, L.D., **Gotsmann, V.L.**, Herkt, C., Ries, F., Kazek, T., Trösch, R., Armbruster, L., Mühlenbeck, J.S., Ramundo, S., Nickelsen, J., Finkemeier, I., Wirtz, M., Storchová, Z., Räschle, M., Willmund, F., (2021), The versatile interactome of chloroplast ribosomes revealed by affinity mass spectrometry. Nucleic Acids Research 49 (1), pp. 400-415. https://doi.org/10.1093/nar/gkaa1192

Rohr, M., Ries, F., Herkt, C., **Gotsmann, V.L.**, Westrich, L.D., Gries, K., Trösch, R., Christmann, J., Chaux-Jukic, F., Jung, M., Zimmer, D., Mühlhaus, T., Sommer, F., Schroda, M., Keller, S., Möhlmann, T., Willmund, F., (2019), The Role of Plastidic Trigger Factor Serving Protein Biogenesis in Green Algae and Land Plants. Plant Physiology 179(3), pp. 1093-1110. https://doi.org/10.1104/pp.18.01252

Trösch R., Barahimibour, R., Gao, Y., Badillo-Corona, J.A., **Gotsmann, V.L.**, Zimmer, D., Mühlhaus, T., Zoschke, R., Willmund, F., (2018), Commonalities and differences of chloroplast translation in a green alga and land plants. Nature Plants 4, pp. 564-575. https://doi.org/10.1038/s41477-018-0211-0

8.10 Curriculum Vitae

Vincent Leon Gotsmann

Education

since 2018 Doctoral student, *Molecular Genetics of Eukaryotes*, RPTU Kaiserslautern

2016 - 2018 Master of Science *Microbial and Plant Biotechnology*, TU Kaiserslautern

2012 - 2016 Bachelor of Science *Biowissenschaften*, TU Kaiserslautern

2003 - 2012 Abitur, Nordpfalzgymnasium Kirchheimbolanden

Profession

since 2024 Scientist *Analytical Development Molecular Biology*, BioNTech Mainz

2023 - 2024 Scientist *Quality Control Cell and Gene Therapies*, BioNTech Idar-Oberstein

2018 - 2023 Pre-doctoral fellow, *Molecular Genetics of Eukaryotes* TU Kaiserslautern

2016 - 2017 Student assistant, *Molecular Genetics of Eukaryotes*, TU Kaiserslautern

2014 - 2015 Student assistant*, IT-Department,* Fraunhofer ITWM Kaiserslautern

Engagement

2019 iGEM Scientific Advisor

2014 - 2018 Member of the student council biology, TU Kaiserslautern

9 Danksagung

Während der letzten sieben Jahre hatten viele Menschen einen Einfluss auf diese Dissertation. Ich möchte Prof. Dr. Felix Willmund für die Betreuung während meiner Promotion danken sowie für die Möglichkeit, an diesem unglaublich interessanten Thema zu forschen. Des Weiteren gilt mein Dank auch Prof. Dr. Thorsten Stoeck für die Übernahme des Zweitgutachtens dieser Arbeit sowie Prof. Dr. Timo Mühlhaus für die Übernahme des Vorsitzes meiner Prüfungskommission.

Ein besonderes Dankeschön möchte ich an Jaro Schmitt und Dr. Günther Kramer vom Zentrum für Molekularbiologie der Universität Heidelberg richten, deren technische Unterstützung diese Arbeit überhaupt erst möglich gemacht haben. Ebenso möchte ich meinen Dank an Prof Dr. Sophia Rudorf und Dr. Nadin Haase ausdrücken, sowie an Dr. Reimo Zoschke und Dr. Michael Ting, welche im Rahmen unseres Mini-Konsortiums wertvolle Beiträge zu dem Erfolg meines Projekts leisteten.

Abgesehen von denen, die direkt in diesem Projekt involviert waren, hatte eine Vielzahl weiterer Menschen einen Einfluss auf meine Promotionszeit und diese Arbeit. Unter diesen möchte ich besonders danken:

Dr. Justus Niemeyer, der niemals müde wurde, jede Art technischer Themen zu besprechen, die während des Laboralltags aufgetaucht sind.

Dr. Anna Probst, deren Bürodekoration (Fotos von Meerschweinchen) uns die schönsten Arbeitsplätze beschert hat, die man sich als Doktoranden wünschen kann.

Dr. Frederik Sommer für die vielen angeregten Diskussionen und unkonventionellen Ideen, sowie für sein exzellentes (F.R.E.D.-) consulting.

Dr. Benedikt Venn für seine Hilfsbereitschaft in allen statistischen und bioinformatischen Belangen sowie für die genialen DIY-Pallettenmöbel hinter dem Labor und die Verwaltung unserer Getränkeversorgung.

Dr. Anna Kiefer für ihren Enthusiasmus für DIY-Projekte wie Hopfen pflanzen und Pilze züchten.

Zuletzt gilt mein besonderer Dank meiner Ehefrau Claudia, die mich mit unendlicher Ausdauer über die letzten sieben Jahre unterstützt und motiviert hat. Vielen Dank für all deine Ratschläge, dein offenes Ohr und deine offenen Arme. Ohne dich wäre diese Dissertation niemals zustande gekommen und ich sehe mich als den glücklichsten Menschen der Welt, weil ich dich in meinem Leben habe!