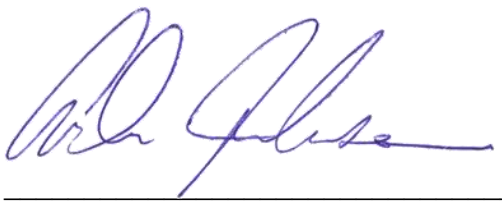
Deep Mutational Scanning Analysis of Ribosomal GTPase LSG1

Completed for the Certificate in Scientific Computation and Data Sciences
Spring 2024

Jordan Risdal
B.S. in Biology (Genetics & Genomics)
Department of Molecular Biosciences
College of Natural Sciences



Dr. Arlen Johnson
Principal Investigator of the Johnson Lab
Department of Molecular Biosciences

Abstract

The protein LSG1 is an essential GTPase involved in ribosome assembly, and is a member of a family of circularly permuted GTPases whose mechanism of activation is poorly understood. To gain better understanding, a deep mutational scanning analysis was conducted to identify key residues in the protein. To do this, two sets of Next Generation Sequencing data produced through PCR mutagenesis were analyzed, one before and one after selection. Mutations which are less prevalent in the dataset produced after selection are presumed to be deleterious, and those residues which have fewer mutations in general in the dataset produced after selection are presumed to be more important. In this way, key residues of LSG1 were able to be identified and further insight gained into the functionality of LSG1. After aligning NGS data onto a reference sequence of LSG1, programs in Python were created to first count mutational variation in both datasets by counting all amino acids found at all residues in the data. Then, counts at too low of a frequency to be considered valid or mutations not theoretically possible through PCR mutagenesis were discarded. Frequencies of each amino acid substitution were calculated, and then fitness scores for each substitution were created, alongside tolerance to mutation scores for each residue. The fitness scores were then visualized in R using ggplot2 to make a heatmap of each substitution's fitness score, the tolerance to mutation scores were mapped onto a preexisting structure of LSG1 using PyMol, and a multiple sequence alignment was created to support results. After visualization, it appears the C-terminus of LSG1 is not entirely important to function, and certain residues such as 182, 184, and 362 that are nearby to G Motifs are likely important to the function of LSG1 and interaction with NMD3, and open further research possibilities.

Introduction

Ribosomes are an essential cellular component in all living things and are responsible for the synthesis of proteins, and subsequently influence growth, reproduction, and the general survival of cells. Therefore, the creation or biogenesis of ribosomes is also incredibly important, and understanding the various factors and proteins that are involved with ribosomal biogenesis can help researchers understand cellular growth and survival in general. One specific enzyme that has been implicated in the nuclear export of the large ribosomal subunit is the protein LSG1, a 640 amino acid long GTPase that has been shown to release the assembly factor NMD3, but functional regions and key residues of which are still unknown (Hedges et. al, 2005). LSG1, like all GTPases, has 6 different motifs which are nearly universally conserved, G1, G2, G3, G4, G5 and a coil motif, all with unique functions related to nucleotide binding and hydrolysis. In order to identify these key residues and understand the impact on the different motifs of LSG1, a deep mutational scanning analysis was conducted in order to identify those residues which are most intolerant to mutation, which substitutions impact cellular fitness the most, and to map mutation tolerance onto a preexisting structure of Lsg1. The work that was done to generate the data that was analyzed in this project was done 5 years ago in the Johnson Research Lab by Daniel Loftus, and involved PCR mutagenesis and subsequent Next Generation Sequencing (NGS). In short, the DNA sequence that encodes for LSG1 was mutagenized using a low fidelity DNA polymerase that introduced various point mutations in the sequence at random intervals. This input DNA before selection was sequenced using NGS, and compiled into FASTQ files collectively called

the initial or init samples. This mutagenized PCR product was then assembled into an expression vector via in vivo recombination into the yeast *S. cerevisiae* in a strain that was a conditional LSG1 mutant under a Gal promoter, the yeast was allowed to grow on glucose media, DNA was extracted and sequenced again, and the sequence results were compiled into FASTQ files, collectively called the product or prod samples. Each of these sample FASTQ files contained reads that were 75 nucleotides (nt) in length, and as result it was important to know the location of each of the reads relative to the sequence as a whole. The rationale behind this experiment is that only those mutations which are tolerated will be found in the initial and product samples, and those that aren't tolerated will be depleted between the initial and product samples. The two groups of FASTQ files, containing the initial/init and product/prod sample sequencing reads are where the work for this project began.

The main goals of this experiment were to quantify the mutations in each of the samples (initial vs product), assign scores to each amino acid substitution that represents the fitness of that substitution when compared to the wild-type amino acid, assign scores to each of the amino acid residues that represents how tolerant each residue is to mutation, create a heatmap of the fitness scores, and map the mutational tolerance scores onto a preexisting map of LSG1. The main prediction for this experiment was that it will be possible to identify the most important residues, but it may be difficult to identify every important residue, due to the slight differences in counts between amino acid substitutions. However, this experiment will provide a good overview of the key areas of LSG1 that are essential and most resistant to mutation and variation and can show the general trends of the functionality of LGS1. Previous experiments in the Johnson lab implicated various residues such as K349 (Hedges et. al, 2005) and S67 as being important, and found that K349 coordinates GTP, while S67 is predicted to sit at the active site channel where it may coordinate water. Substitutions in these residues such as S67A were found to but tolerated, while other substitutions such K349N and S67R were found to be lethal. Therefore, another prediction is that the results of this experiment will be able to support the previous findings from in vivo molecular biology and genetics work. In conclusion, this project seeks to identify the key residues of the GTPase LSG1, and will do so by analyzing preexisting NGS data in Python and R in order to quantify mutations, assign fitness scores to substitutions, assign tolerance scores to residues, and visualize these scores, with the expected outcome of being able to identify the most important residues and general areas important to the function of LSG1.

Materials & Methods:

Mapping to Reference Sequence

Initially, this project was started by trying to follow the pipeline set forth in the paper *A single 2'-O-methylation of ribosomal RNA gates assembly of a functional ribosome* (Yelland et. al, 2023), previously written in Dr. Johnson's lab, which employed a similar deep mutational scanning analysis in order to identify key residues of the protein NOG2, another ribosomal assembly factor that is closely related to LSG1 and is also circularly permuted. This pipeline involved using the program BowTie2 to align the sequence reads onto a reference protein sequence, and then using the program TileSeq in order to split the sequences into separate tiles

and count mutations there, and subsequently output fitness scores and tolerance scores. I was initially successful, and the sequence reads in the FASTQ file were successfully mapped onto a reference sequence of LSG1 using the program BowTie2. BowTie2 is a program that is used in a Linux based environment with command line coding, which was substantially difficult at first but was relatively straightforward. The output of this program is a SAM file or Sequence Alignment Map file, which is a tab delimited text file that contains each read, where in the alignment sequence the read was aligned to, and how accurate each alignment is. Another Linux based program called samtools was then utilized, which is able to modify SAM files and filter out all reads that were below a certain threshold of mapping quality, in order to eliminate reads that were inaccurately mapped or not aligned at all. This filtered file was the file that was eventually used to count mutations, and what I attempted to use the TileSeq program on. Next, TileSeq, also a Linux based program, was employed to try to further follow the preexisting pipeline. The installation of the program was difficult, but the main difficulty encountered was that the program utilized a parameter sheet that specified all of the experimental conditions and had very precise requirements. This program was created by a specific lab in order to accomplish a specific task, and as a result many of the requirements of the parameter sheet did not apply to this project, and only complicated the process. Many lesser-known bioinformatics programs have very specific functions and limited use outside of the task and lab that they were designed for, and as a result it was difficult to apply them to this unique project. Other programs used for deep mutational scanning analyses were experimented with, such as Enrich2, but they were still not relevant to this project. After several unsuccessful attempts with other programs, I decided to deviate from the preexisting pipeline and instead independently create programs in Python that would accomplish the goals set forth in the introduction.

Counting Mutational Variation

The most important goal and the first main step was to count the mutational variation present in each of the samples and subsequent SAM files produced with BowTie2. In short, to do this a program in Python was created that iterated over each alignment read in the SAM file. The program checked if the alignment contained the codon of interest, and if it did, it counted the amino acid that was present in that codon in that read by adding it to a dictionary of counts for that specific codon. The alignment SAM files were essential in doing this, as the information about where in the sequence that read is aligned is present in the files. After every read was iterated over, checked if it contained the codon of interest, and the amino acid present at that codon was counted, the dictionary containing the counts for that specific codon was added to a list that contained all of the different dictionaries for the codons. This process was then repeated for the next codon, and repeated for every other codon in LSG1, of which there are 640, however only 639 were counted. The start codon was skipped in this analysis, as it is assumed that any protein sequences that have a start codon mutated will be nonfunctional and therefore shouldn't theoretically exist. Finally, this list containing the dictionaries of codon amino acid counts was exported to a .csv file, which was a 20x640 table that contained columns for each amino acid and rows for each codon, with each cell corresponding to the number of times that amino acid was counted at that codon in all of the reads. This process utilized for loops in order to iterate over each codon in each read, as this was found to be the simplest way to count mutations with the programming knowledge possessed. Combined, the initial and product alignment files were

16Gb in size, which resulted in the program taking multiple hours to run. It's also possible that this slow run time was due to the reliance on for loops in this first program, and if the process had to be repeated many times, I would employ list comprehension more to shorten run time. However, it's also fair to assume that the long run time is due to the massive size of the alignment file size and sequencing data size in general, so all blame can't be placed on the program itself. Regardless of run time and methods employed in this program, two .csv files that contained counts of each amino acid at each codon in both the initial and product files were now the output. However, this output file was still unclean and further work was needed to get rid of those reads that were due to sequencing error.

Cleaning Data

One of the main issues with the output count data was that there was a fair amount of sequencing error that resulted in amino acid counts that were very low in number; some had counts of approximately 10 while other amino acid counts for that codon contained approximately 10^6. These small counts could not be included in the final analysis and scoring of amino acid substitutions due to their ability to confound results. There were also many counts that were theoretically impossible through PCR mutagenesis, a process explained in the introduction. This experiment assumes that PCR mutagenesis is only able to modify one nucleotide at a time, therefore when amino acids are mutated in this way, it is assumed it's only possible for it to mutate to an amino acid that is vertical or horizontal to the original amino acid on the codon chart. Because of this fact, there was a limited number of amino acids that were theoretically possible to mutate to through PCR mutagenesis, and any counts of those amino acids at those codons were likely due to sequencing or mapping errors. In order to make downstream analyses more accurate, these counts would have to be removed. The most important reason for doing this was that it was planned to create fitness scores based on the frequency of counts in the initial vs the product sample, and if low frequency counts were kept in, any slight change would be overrepresented and confound the results by suggesting that those substitutions have more or less fitness than in actuality. Another Python program was created in order to accomplish this, which iterated over each row in the .csv file using for loops again. The program checked each amino acid count against a dictionary of theoretically possible mutations at that codon, and checked to see if it was at a higher frequency than 0.0005 for that codon. This frequency was decided on by looking at the approximate frequency that nonsense mutations were observed, and chosen because it removed most nonsense substitutions and many of the smaller counts. If a substitution wasn't considered theoretically possible, or it was at too low of a frequency, that cell was changed to zero, and the next row was checked. By doing this, it was possible to clean the data and utilize only those counts that were theoretically possible and at a high enough frequency to be statistically relevant.

Calculating Frequencies and Fitness Scores

The next goal was to create fitness scores for each amino acid substitution at each codon. To accomplish this goal, a Python program was created that made a .csv file that contained the proportion of counts that were of that amino acid substitution in the product over the initial samples. The proportions of the amino acid counts were used to make these scores, because there obviously slight differences in total counts due to sequencing error, and depth of sequencing at

different parts of the sequence. Because of this, it was more important to know if an amino acid substitution changes as a proportion of the total counts at that specific residue. If there were significantly fewer counts of an amino acid and a subsequently lower proportion, then the fitness score will be lower than one, those that have about the same amount in both the product and initial will have a score of approximately one, and any that have a larger proportion in the product than the initial samples will be larger than one. However, it is assumed that very few will be larger than one. The proportion of each amino acid substitution in the product sample and in the initial sample was calculated and compiled into two separated lists, and these two lists were divided, with the product over the initial, making sure to avoid division by zero as well, and then output to another .csv file.

Calculating Tolerance Scores

The final score that was made that quantified the sequencing results was another .csv table that measured each position's tolerance to mutation. This was calculated by calculating the proportion of the wild-type amino acid in the product sample over the initial sample. The wild-type amino acid is assumed to always be the one with the highest count, and this is proven to be true when comparing to the wild-type sequence. It is assumed that those which have a larger proportion of wild-type amino acid in the product sample will be less tolerant to mutation, and those that have a lower proportion of wild-type amino acid in the product sample will be more tolerant to mutation. These tolerance scores ranged from approximately 0.99 to 1.03, which are relatively similar numbers, however, since the number of amino acid counts is in the millions, it's fair for the numbers to be more precise. This table of proportions was also output to another .csv file.

Visualizing Results

Now that all mutations were counted, all erroneous data discarded, and all fitness scores and tolerance scores calculated, the final step was to visualize the resulting data. There were two main ways in which the data was to be visualized, in a heatmap that visualized the fitness scores for each amino acid substitution at each codon, and by mapping the tolerance to mutation scores onto a preexisting structure of LSG1 in order to visualize those areas that are most intolerant to mutation. In order to create the heatmap, the programming language R was utilized, and specifically the package ggplot2 was used in order to create the resulting heatmaps (Wickman et. al, 2016). R was chosen to visualize the data, as it is a powerful tool for visualizing data, and has a wide range of customizability that lends itself well to bioinformatics visualizations. The fitness scores created earlier were loaded into R, and some data wrangling and cleaning was performed in order to make it easier to work with, such as changing all 0 values into NA, specifying the wild-type amino acid sequence, and 'melting' the data into a long form table in order to visualize it better. Finally, the geom_tile() function from the ggplot2 package was utilized in order to display a heatmap for each amino acid substitution score at each codon, with the x axis labeled with the wild-type amino acid at that position, and the position itself. Green tiles represent those substitutions that have a high fitness with a gradient of colors processing down to red to represent those with a low fitness. Heatmaps were made for every position in LSG1, as well as a heatmap that specifically targeted the residues present in the GTPase motifs.

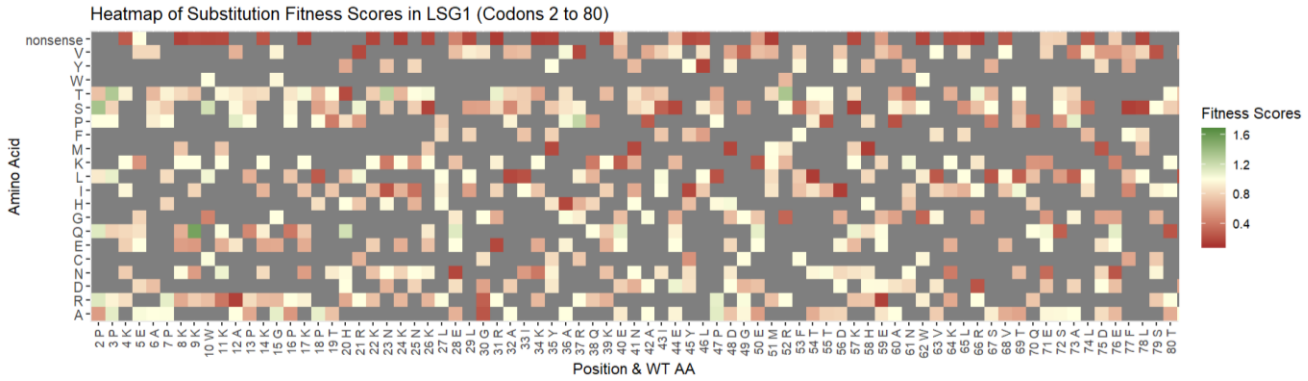
In order to map the tolerance to mutation scores onto a preexisting structure of LSG1, the program PyMol was utilized, which is a protein visualization tool that utilizes Python as a base programming language in order to interact with and modify protein structures. A preexisting structure of LSG1 was used that was predicted using AlphaFold, an AI program that estimates protein structure based on predicted biochemical interactions between residues (Jumper et. al, 2021). This structure is sufficient, but there are some regions that aren't properly predicted, such as the N-terminus, and further studies and cryo-electron microscopy would be needed in order to make the results more valid. Regardless, this structure was loaded onto PyMol, and a Python script was made that modified heat values in the protein structure, which can be changed to represent a number of different values, into the tolerance to mutation scores. The scores were mapped onto the structure using color, with white representing higher values, and a gradient processing to red representing lower values.

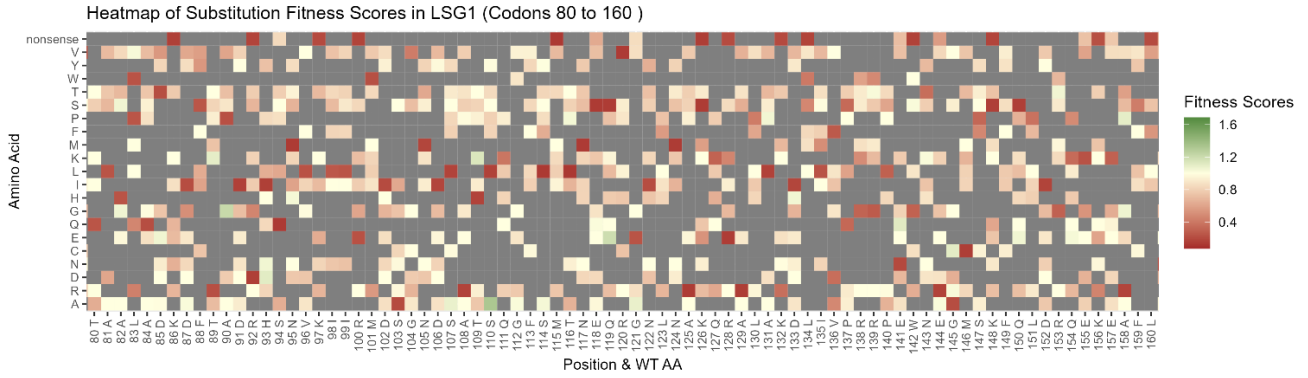
At the suggestion of the SCDS committee, a multiple sequence alignment of LSG1 was made and visualized in order to try and support the results found from the deep mutational scanning analysis, and to see if any of the key residues that were identified as being intolerant to mutation were also highly conserved among many species. The multiple sequence alignment itself was done by using sequences of LSG1 taken from the NCBI database for *S. cerevisiae*, *H. sapiens*, *M. musculus*, *G. gallus*, and *D. rerio*. These species were chosen as they represent yeast, the model organism that was used in this analysis, and a number of different vertebrates including humans, of which the implications for are more important. These sequences were then put into the program TCoffee, an online multiple sequence alignment tool run by the Centre for Genomic Regulation of Barcelona (Chang et. al, 2015), which ran a multiple sequence alignment, and then the online program Esprit was used in order to visualize the resulting alignment and highlight those residues that are and aren't conserved (Robert et. al, 2014). With this final figure visualized, all methodology for the project was completed.

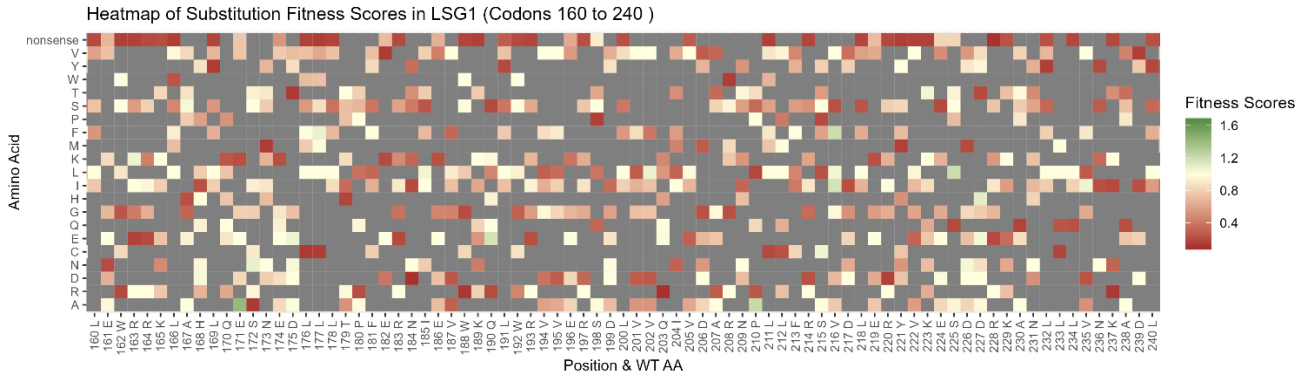
Results & Interpretations:

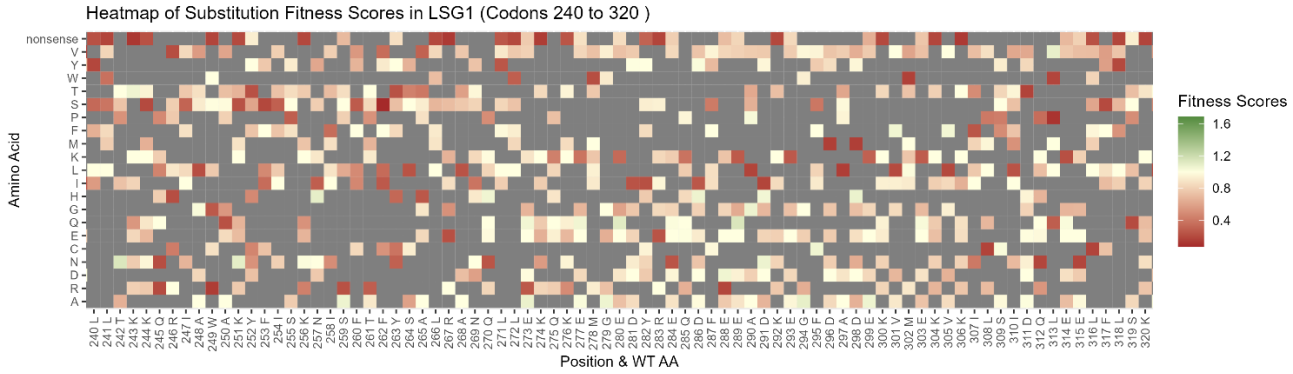
The interpretation of results can be better determined by breaking up the interpretation by figure. First looking at the heatmaps present in Figure 1, as well as the data in Table 1 that supports Figures 1 by showing the most and least fit substitutions, some general trends are able to be ascertained by looking at the total heatmap and heatmap of the GTPase motifs. One important result found in the complete heatmaps is that the C terminus of the protein is seemingly tolerant to mutation, as essentially all substitutions towards the end of the protein have a neutral or even slightly positive fitness score, even nonsense mutations. The nonsense mutations are of particular note, as those mutations would have prematurely truncated the protein, and if those mutations are tolerated, it indicates that the C terminus is seemingly not very important. This was initially believed to have been an artifact in the results due to the low sequencing read depth at the ends of the protein, however, if this was the case you would also expect to see a similar pattern on the N terminus, but none is seen. The sequencing primers that were utilized were also far enough away from the open reading frame to suggest that there weren't any artifacts from the experimental design, so this result is assumed to be genuine. There are also almost no substitutions which have a positive fitness score, which is expected, as since

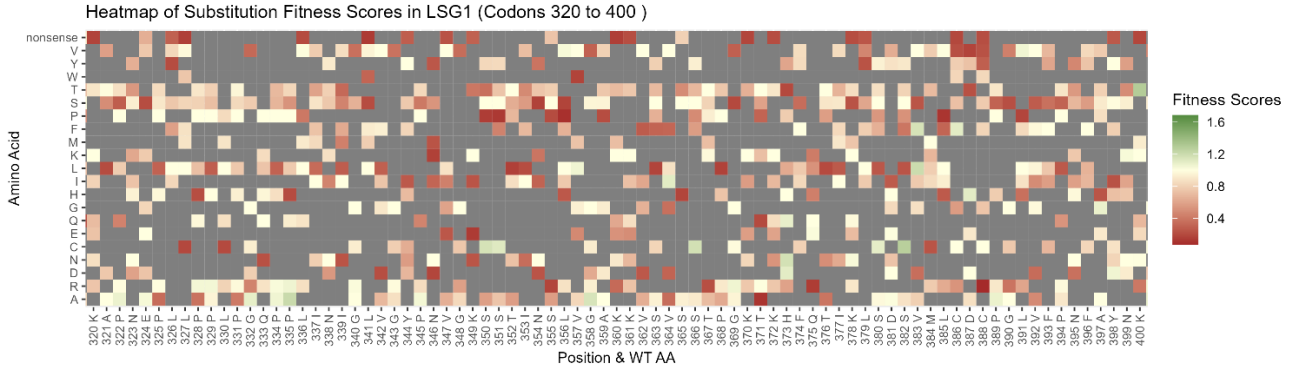
LSG1 is a fairly important ribosomal GTPase, any deviation from the sequence and subsequent structure would result in reduced growth. Only a few substitutions had any positive impact, such as P3T, K9Q, D453A, and E171A. These specific residues may enhance the ability of LSG1 to interact with NMD3 or hydrolyze GTP, but that is only speculative at this point. Those substitutions that had the most negative effects and therefore least fitness scores were found to often be in one of the canonical GTPase motifs, such as C388R and T371A. However, other substitutions were found to also be not tolerated and yet aren't a part of the GTPase motifs, such as N184D, L356P, and L385P. These substitutions in particular are interesting though, as while they are not directly in any motif region, they are very close by to one of them, indicating that this residue may be interacting with the motif in a positive manner, or assisting in the GTPase activity in a way that is not fully known.

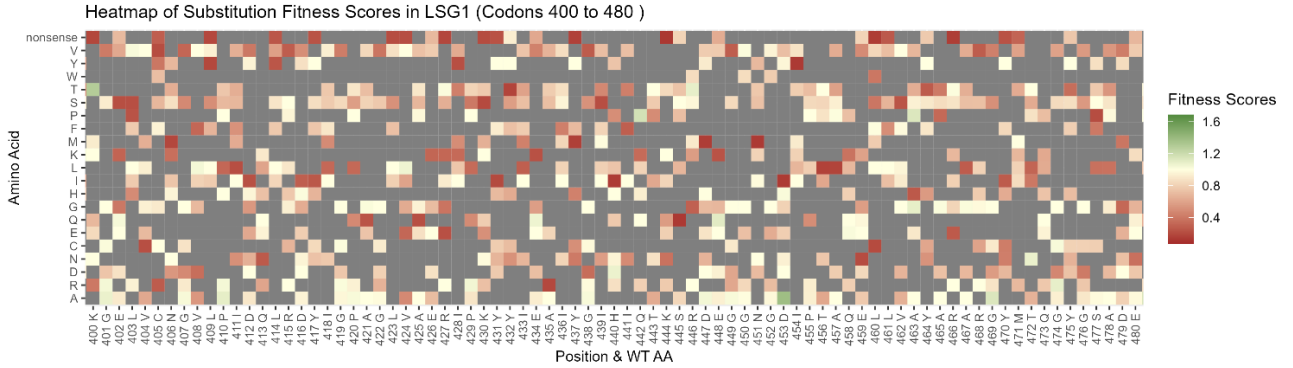


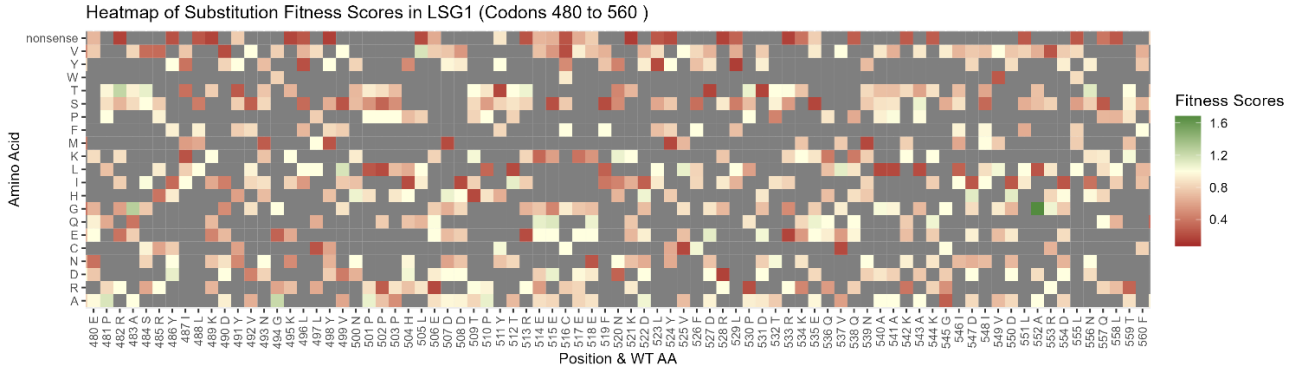





Heatmap of Substitution Fitness Scores in LSG1 (Codons 240 to 320)
Amino Acid
nonsense
Position & WT AA
Fitness Scores
1.6
1.2
0.8
0.4


Heatmap of Substitution Fitness Scores in LSG1 (Codons 320 to 400)
Amino Acid
nonsense
Position & WT AA
Fitness Scores
1.6
1.2
0.8
0.4


Heatmap of Substitution Fitness Scores in LSG1 (Codons 400 to 480)
Amino Acid
nonsense
Position & WT AA
Fitness Scores
1.6
1.2
0.8
0.4


Heatmap of Substitution Fitness Scores in LSG1 (Codons 480 to 560)
Amino Acid
nonsense
Position & WT AA
Fitness Scores
1.6
1.2
0.8
0.4

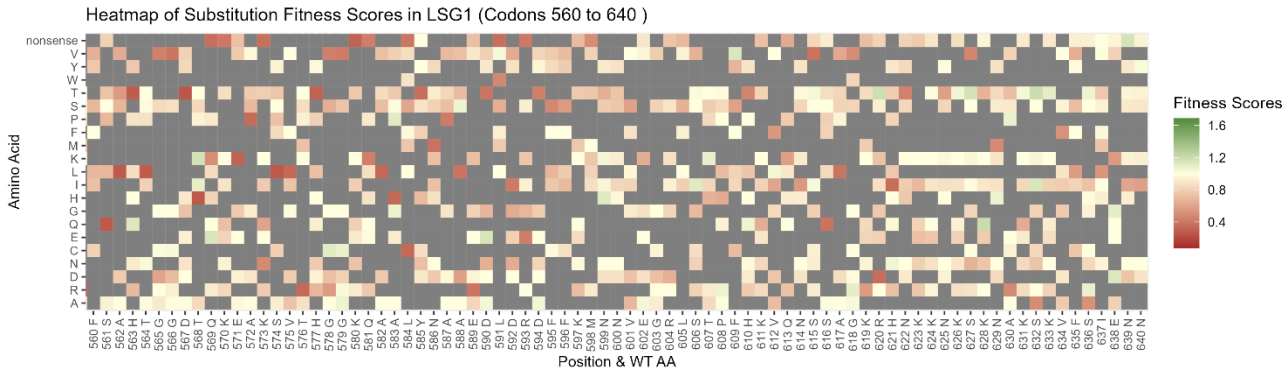


Figure 1: Heatmaps of Amino Acid Substitution Fitness Scores at all positions in LSG, green being more fit, red being less fit.

By examining the heatmap that specifically looks at the GTPase motifs in Figure 2, other trends can be determined and previous experiments supported. The most noticeable thing about this smaller heatmap is that essentially all substitutions have a low fitness score and almost all of the nonsense substitutions have a low fitness score. This result is able to validate the experiment, as all nonsense mutations shouldn't be tolerated as they result in a truncated protein. The only substitution in any of the motifs that had a high fitness score was E171A, which was previously mentioned. Other residues were previously identified in the Johnson lab to be essential, such as 349, and mutations to that residue such as K349N and K349R in vivo resulted in decreased cell viability. This heatmap is able to support those findings, as those same substitutions are shown to have a low fitness score. This figure provides various other benchmarks through which the results found in this project can be validated.

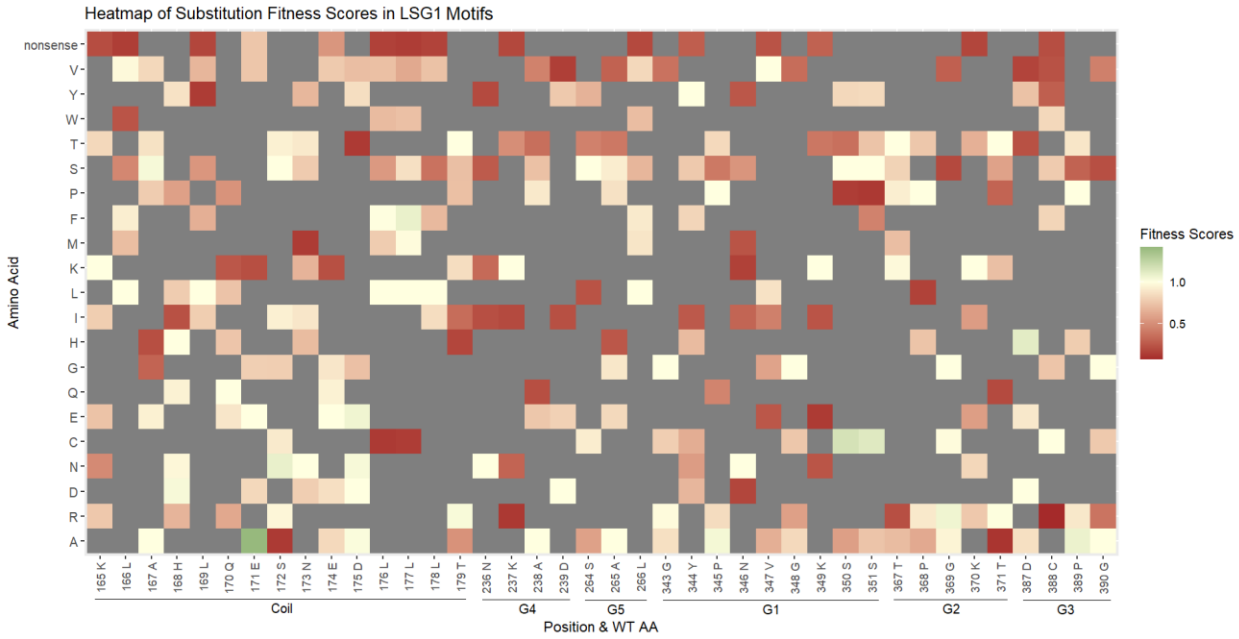


Figure 2: Heatmaps of Amino Acid Substitution Fitness Scores specifically in the GTPase motifs of LSG1

Looking at the tolerance to mutation scores that were mapped onto a preexisting structure of LSG1 in Figure 3, as well as looking at Table 2, which contains the residues most and least tolerant to mutation, we are able to see other important pieces of information. What is most easily visible is the C terminus of the protein, which is in red and is seemingly intolerant to mutation. As previously stated, this was initially believed to be an artifact from the sequencing itself, but this is now believed not to be the case. The fact that the mapping shows an intolerance to mutation in the C terminus is puzzling, as it directly contrasts the information found in the heatmaps in Figures 1 and 2, which show that nonsense mutations in the C terminus of LSG1 are relatively fit. However, it does not completely confound the findings found previously, as its only approximately the last five amino acids of the C terminus which are intolerant to mutation, while the heatmaps suggest that the last forty amino acids of the C terminus are not as important. This may just suggest an alternate role for the very end of the protein, rather than the C terminus as a whole. However, what is more likely is that this is just an unexplained artifact in the experimental design, and is due to the very low read depth at the ends of the protein. Other results found in Figure 3 and Table 2 are able to validate this experiment by supporting previous work in the Johnson lab, as the residues 349 and 346 were found to be quite intolerant to mutation, and previous experiments in vivo showed that mutations to these residues are deleterious. Other residues were found to be particularly intolerant to mutation and also be in close proximity to the GTPase motifs, such as 184, 182, 360, and 249. This suggests that these residues may be important in assisting in each motif's functionality or have a unique currently unknown function.

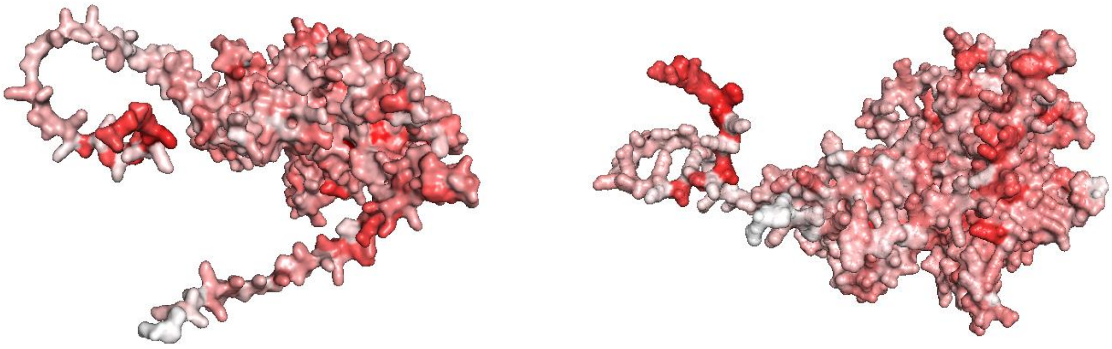


Figure 3: AlphaFold Predicted Structure of LSG1 with Tolerance Scores mapped to b values, white being more tolerant to mutation, red being less tolerant to mutation. Two different angles shown, the second one being rotated down 90 degrees from the first.

Finally, examining the multiple sequence alignment created in Figure 4 allows the previously discussed results to be supported and provide further insight. Regarding the C terminus, it is not largely conserved in all of the five organisms selected, indicating that it is most likely right to assume that the C terminus is largely not very important. However, residue 629 is conserved, and was also found to be quite intolerant to mutation, so it may be that that specific residue of the C terminus is important, while that area in general isn't. Figure 4 also supports the idea that residues adjacent to the GTPase motifs are important, as most of those residues adjacent to motifs are highly conserved. This supports the idea that residues such as 182, 184, 249, and

360 are important due to their proximity to the motif regions, as all of those residues are also highly conserved. In general though, it is difficult to make many claims about the results of Figure 4 without also supporting them with results from the other figures.

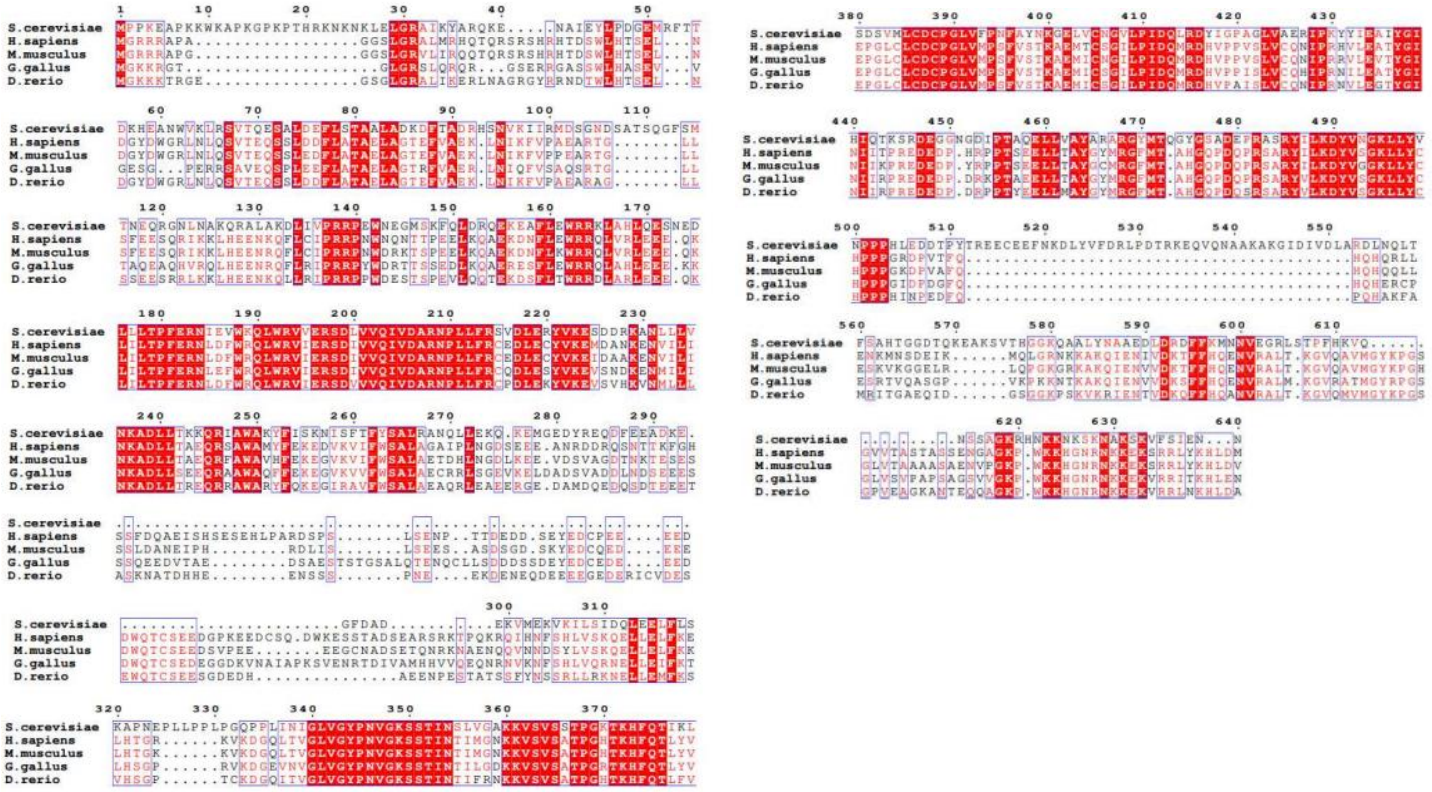


Figure 4: Multiple Sequence alignment of LSG1 in five different organisms, with position on top and conserved sequences highlighted in red

Ten Most Fit Amino Acid Substitutions			Ten Least Fit Amino Acid Substitutions		
Substitution	Motif	Score	Substitution	Motif	Score
A552G		1.68651	C388R	G3	0.07215
K9Q		1.54836	F262S		0.07341
E171A	Coil	1.41208	L356P		0.09279
D453A		1.38057	Q203R		0.09618
P2S		1.3478	N184D		0.09715
R52T		1.33814	R228(nonsense)		0.09897
S110A		1.32662	L313P		0.10877
P3T		1.31287	L385P		0.1095
K400T		1.2943	T371A	G2	0.10952
A483G		1.27985	K243(nonsense)		0.12035

Table 1: Ten Most and Ten Least fit Amino Acid Substitutions

Ten Residues Least Tolerant to Mutation				Ten Residues Most Tolerant to Mutation			
Position	Wild-Type AA	Motif	Tolerance Score	Position	Wild-Type AA	Motif	Tolerance Score
349	K	G1	1.03414	3	P		1.00046
346	N	G1	1.03126	2	P		1.00061
184	N		1.02972	112	G		1.00132
634	V		1.02835	7	P		1.00149
635	F		1.02826	450	G		1.00157
437	Y		1.0282	294	G		1.00179
629	N		1.02694	15	G		1.00212
360	K		1.02668	279	G		1.00217
249	W		1.02658	603	G		1.00235
182	E		1.02599	331	P		1.00254

Table 2: Residues most and least tolerant to mutation and their wild-type amino acid

Discussion:

With all resulting data visualized and interpreted, the final step in this project is to discuss the importance of the results and what they mean in the context of all of the other work done in this experiment and in the Johnson Lab in previous years. One of the main results that should be discussed is with regards to the C terminus. As previously discussed, the heatmaps in Figure 1 suggest that approximately the last forty amino acids in the C terminus of LSG1 are unimportant, due to the nonsense substitutions in those regions being relatively fit, and all substitutions in those regions also having a relatively high fitness. This suggests that the C terminus is largely unimportant to the function of LSG1, and also supports previous experiments done in the Johnson Lab where the C terminus of LSG1 was truncated, and impacted on cell growth, but the cells were still viable. However, the results found in Figure 3, which suggest that approximately the last five amino acids are intolerant to mutation, seem to conflict with those results. However, upon further reflection, it is believed that the C terminus is still not especially important. This is due to the fact that a large portion of the C terminus was found to be unimportant from Figure 1, while only the very end of the C terminus is suggested to be important from Figure 3. While it was previously stated that this result is likely not due to any artifact from the sequencing process, it is possible that the low sequencing read depth at the very end may have influenced the results found in Figure 3 due to it being at the very end of LSG1. The results found in Figure 4 also support the notion that the C terminus is not entirely essential to the function of LSG1, as the residues in that region are not highly conserved. If this experiment were to be repeated, it would probably be best to sequence LSG1 in a way that had overhangs of 75nt on the ends of the protein, and to align the reads onto a reference sequence that also had 75nt overhangs in order to have equal read depth throughout the entire protein. Despite some confounding information, the C terminus in LSG1 still seems to be nonessential.

Another main result that should be discussed is the prevalence of seemingly important residues directly adjacent to the canonical GTPase motifs. Residues such as 182, 184, and 360 were all found to be highly intolerant to mutation, and each of these positions are within a few

residues of the Coil and G2 loop, respectively. The residues 182 and 184 are of particular note, as predicted structures of LSG1 and NMD3 have suggested that the region of 160-190 in LSG1 seems to interact with NMD3, and so those residues may help coordinate interactions with NMD3. Other substitutions in specific residues, such as L385P and F262S also have very low fitness, indicating that those residues are also quite important for the functionality of LSG1. While it is possible that the prevalence of intolerant mutations nearby the GTPase motifs is purely coincidental, the frequency with which they are found makes it likely that these residues are functionally important. These residues may be acting to support the main function of the nearby GTPase motifs by directly helping with nucleotide binding and hydrolysis, or they may be supporting that main function in some unknown auxiliary way. Further experiments could focus on these residues and identify exactly what they do and their importance in vivo.

Other results that should be discussed are with regards to the general experimental design, and how that design may have impacted the results. As stated in the introduction and methodology sections, this project didn't have any input into the creation of the data, so if it were to be repeated, some steps would be altered, such as first transforming the initial PCR product into yeast and allow it to grow, then selecting against it and obtain to product sample in order to have more equal data sets. As previously discussed, it's also possible that the results surrounding the C terminus may have been impacted by the sequencing methods, and so if this was repeated, LSG1 would have sequencing and mapping overhangs in order to have equal read depth. One result that should also be discussed that may be due to the experimental design is that all of the residues that are most tolerant to mutation have either Proline or Glycine as their wild-type amino acid. As these amino acids are both hydrophobic in nature, it may suggest that in LSG1 those kinds of amino acids are most unimportant in the functionality of the protein. It may also be that, since there are many hydrophobic amino acids, it is simply more likely that those residues will mutate to a biochemically similar amino acid, and therefore have little impact on the function of LSG1. However what is most likely believed to be the cause of this anomaly, is a slight preference for mutation to specific nucleotides by the polymerase used in PCR mutagenesis. Overall though, it is difficult to blame any results on the experimental design, and it is more informative and helpful to try to understand the results rather than explain them away.

One final way that the results should be discussed is with relation to previous experiments done in vivo in the Johnson lab. By comparing the results found in this project to previous validated results in other experiments, it can be possible to verify and validate the results found here. One experiment that was done previously was to truncate the C terminus of LSG1 and observe resulting growth. As previously discussed, this project found that the C terminus is likely unessential to the function of LSG1, and this is supported by these previous experiments, as there was little effect on growth when the C terminus was truncated. Other experiments were involved with specific point mutations that mutated a single amino acid in the sequence of LSG1. Substitutions such as N173Y, R267G, K349N, K349R, S350I, and C388R were all made in vivo LSG1 previously, and all resulted in dominant negative or complete loss of function phenotypes, indicating that they were essential residues. These same residues, and in particular 349, were found to be highly intolerant to mutation, indicating the experimental design is valid. One previous experiment that was discussed in the introduction was the substitution

S67R, which was found to be a deleterious substitution, and S67 was expected to be an important substitution. This was one experiment that was not validated, as S67R was not theoretically possible to be obtained through PCR mutagenesis, and so other experiments would have to be done to investigate that residue. While the data isn't able to investigate that specific hypothesis, there are various benchmarks by which to verify the results of this project, and which help to confirm the validity of these results.

In conclusion, this deep mutational scanning analysis done on NGS data of the ribosomal GTPase LSG1 was able to verify its results through the comparison to previous experiments, identify the C terminus as nonessential in LSG1, and identify various residues that are adjacent to GTPase motifs that may be essential to the functionality and stability of LSG1. All of these results allow for better understanding of LSG1 and GTPases, and open up many further possibilities for experimentation and deeper understanding of LSG1 and ribosomal biogenesis as a whole.

Acknowledgements:

I would like to thank Daniel Loftus for producing the dataset that was analyzed in this experiment, Dr. Johnson for his help in guiding my experiments, thought process, and writing throughout this project, Ran Lin for her insight into the results, and all other members of the Johnson lab for their input and discussion around the methodology and results of this project.

Literature Cited:

Chang, J. M., Di Tommaso, P., Lefort, V., Gascuel, O., & Notredame, C. (2015). TCS: a web server for multiple sequence alignment evaluation and phylogenetic reconstruction. *Nucleic acids research*, *43*(W1), W3–W6. https://doi.org/10.1093/nar/gkv310

Hedges, J., et al. "Release of the export adapter, NMD3P, from the 60S ribosomal subunit requires rpl10p and the cytoplasmic gtpase LSG1P." The EMBO Journal, vol. 24, no. 3, 20 Jan. 2005, pp. 567–579, https://doi.org/10.1038/sj.emboj.7600547.

Jumper, J. et al. "Highly accurate protein structure prediction with AlphaFold." Nature, 596, pages 583–589 (2021). DOI: 10.1038/s41586-021-03819-2

Robert, X. and Gouet, P. (2014) "Deciphering key features in protein structures with the new ENDscript server". *Nucl. Acids Res*. **42**(W1), W320-W324 - doi: 10.1093/nar/gku316 (freely accessible online).alThe PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC.

Wickham H (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

Yelland, J.N., Bravo, J.P.K., Black, J.J. et al. A single 2′-O-methylation of ribosomal RNA gates assembly of a functional ribosome. Nat Struct Mol Biol 30, 91–98 (2023). https://doi.org/10.1038/s41594-022-00891-8

Computer Code:

All code, excluding code that was run in a Linux based environment and therefore cannot be saved as a file, was compiled into a public GitHub repository and can be found at this link:

https://github.com/jrisdal/LSG1-SCDS-Project

Reflection:

By working on this project throughout the last few months, I have been able to learn many things and actually do real work in biological computation and bioinformatics analysis in general, which has been invaluable in providing me with experience and allowing me to recognize what my interests and career goals are in the field of biological research. First and foremost, I have gained a lot of experience coding and working with various programs. I had to learn how to use Linux based programs such as BowTie2 and samtools in order to begin the analysis, which was something I was not very familiar with as it is quite different from interface-based programming. However, being able to learn this has given me more opportunities to do bioinformatics work in the future, as many advanced bioinformatics programs are Linux based. I was also able to apply previous theoretical knowledge I learned in various programming classes in a hands-on way. While I was able to code in R and Python before beginning this project, I was able to apply my knowledge to an actual project and gain much more experience in optimizing my code, properly annotating and documenting it, and identifying the best strategy for achieving a goal through programming. I've also been able to modify, clean, and wrangle my own data in order to visualize it in R using ggplot2, which is something I had only ever done with datasets provided to me. The ability to do this will allow me to work on projects in a much more experienced manner in the future, and identify exactly what I need to do for a project and how I should go about it. I was also able to learn how to effectively write scientific papers by writing up this report. I have written various reports for classes before, but often they were just theoretical in nature, and involved little discussion of results, or use of independently developed methods. Throughout this process, I have had to critically think about the resulting data, and how the methods I employed may have impacted the results. Overall, I have gained so much experience in terms of biological computation, and actual scientific research and reporting, which has made me a better candidate for opportunities in the future, and a better scientist.