Group 2 Final Report: Investigating the Evolution of SARS-CoV-2 through Codon Bias Analysis

April Batts, Amber Ginn, Mary Hubley, Jenna Green, Kutluhan Incekara, & Hoff Lindberg

University of Maryland Global Campus

BIOT 670: Capstone in Biotechnology

Dr. Wolfgang Rumpf

March 29, 2022

Table of Contents

Executive Summary

The primary goal for this project is to look for evidence of rapid evolution in the genome of the Wuhan-Hu-1 strain of the SARS-CoV-2 virus through codon usage analysis. SARS-CoV-2 is the virus responsible for the COVID-19 pandemic, and the Wuhan-Hu-1 strain is the original strain of this virus. Its exact origin and evolutionary history are still unknown, and the aim is to investigate whether rapid evolution was possibly involved, either through cross-strain hybridization or laboratory engineering, and whether any evidence of such could be found in the codon bias of this strain. In pursuit of this goal, our approach was first to develop a python script that will calculate the codon usage bias in the Wuhan-Hu-1 genome, plot this data graphically using R, and compare it against the same analysis performed with other coronaviruses, or CoVs. Through this analysis, our findings largely are in agreement with prior studies. Here, we report that while similarities in RSCU distribution are observed between CoVs from different host species, some observable variation exists. Notwithstanding, the codon usage bias of SARS-CoV-2 appears most similar to the non-human pathogenic CoVs investigated in this analysis. This includes RaTG13, BANAL-20-52, and pangolin CoV. Wuhan-Hu-1, however, differed in codon usage for Cysteine specifically in the membrane protein gene. When examining ORF1ab, Wuhan-Hu-1 was found to have fewer overly expressed codons (RSCU > 1.5) when compared to BANAL-20-52, HCoV-OC-43, and RaTG13. Very little difference was observed between Wuhan-Hu-1 and other strains of SARS-CoV-2.

Introduction & Background

History of SARS-CoV-2 and Pandemic

All living things continuously undergo evolution and natural selection; this is particularly true of viruses, although they require a host for survival and thus may not qualify as "living" on their own (NatureEducation, 2010). A virus's purpose is to find a host where it can replicate itself while evading the immune system and then spread to new hosts. There are specific characteristics that will help a virus to accomplish this and then others that are not as beneficial. These mutations will occur over time, often frequently, to provide viruses with a high capacity for adaptation. As this evolutionary pressure takes place, there are certain codons of a virus' genome that are preferentially used over others. The genetic code consists of 64 codons which translate into 20 amino acids; therefore, multiple codons encode for the same amino acid (NatureEducation, 2014). A particular virus often uses some redundant codons more frequently than others. This frequency found in the genome of an organism is referred to as codon usage bias. Factors such as translation rate and natural selection affect the usage of codons in species. Using this method of analyzing codon usage distribution is a way to differentiate between viral strains.

Coronaviruses (CoVs) are taxonomically placed under the family *Coronaviridae* and subfamily *Coronavirinae.* They are the positive-stranded RNA viruses that have been known for decades. Coronavirus infections in humans were related only to mild respiratory diseases until the 2000s. However, Severe Acute Respiratory Syndrome CoV (SARS-CoV) and Middle East respiratory syndrome CoV (MERS-CoV) outbreaks took many lives in 2002 and 2012,

respectively. These events showed that newly emerged CoVs had become a serious threat to human life (Kadam et al., 2021).

In December 2019, a coronavirus began spreading among humans in Wuhan, China which has led to the current global pandemic fueled by the virus named “severe acute respiratory syndrome coronavirus 2” (SARS-CoV-2) (Singh & Yi, 2021). The evolution of the Wuhan-Hu-1 strain of SARS-COVID-19 has not been fully characterized; therefore, assessing whether rapid evolution or laboratory engineering has occurred has become a major point of focus when researching the virus. Understanding the continuous evolution of SARS-CoV-2 provides information for mitigating disease transmission, treatment of COVID-19 infections, and preventing future outbreaks.

SARS-CoV-2 Evolutionary Relatives and Other Similar Coronaviruses

Literature was reviewed to understand the evolutionary relationship between SARS-CoV-2, Wuhan-Hu-1, and other coronaviruses so candidates of other viral strains could be selected and applied to our analyses for comparison to the Wuhan-Hu-1 strain. Hundreds of peer-reviewed articles have published various findings on the emergence of SARS-CoV-2. An article by Li, Lai, Gao, and Shi (2021) provides a detailed summary of how this research has progressed over the prior two years with new close relatives of SARS-CoV-2 being identified over time. Current research has demonstrated the most closely related species of coronaviruses to the Wuhan-Hu-1 strain are those isolated from bats in Laos despite having a geographic origin dissimilar to where the original SARS-CoV-2 strain was isolated in Wuhan, China (Mallapaty, 2021). However, the bat CoV RaTG13 also has a high sequence similarity (96.2%) to SARS-CoV-2 (Li, et al., 2021) with this strain being discovered in 2013 from bat droppings in

Yunnan, China. RaGT13 has been applied to many studies for comparison with SARS-CoV-2, including at least one analysis specifically on codon usage patterns (Hou, 2020). Nonetheless, the most recently discovered, closely related bat CoVs are those identified in Laos, though the research is in pre-publication (Mallapaty, 2021). BANAL-20-52, one of the strains identified in Laos, was reported to be the closest relative of SARS-CoV-2 with 96.8% sequence similarity. Therefore, RaTG13 and BANAL-20-52 are strong candidates to include in our comparisons to Wuhan-Hu-1's codon usage while investigating how it may differ from its closest known relatives, if at all.

For comparison against other human-pathogenic viruses, there are only seven coronaviruses known to infect humans, known as hCoVs, including SARS-CoV-2 (Schoeman, Gordon, & Fielding, 2021). The other six include four other betacoronaviruses SARS-CoV, MERS-CoV, HCoV-OC43, and HCoV-HKU1, and two alphacoronaviruses HCoV-229E, and HCoV-NL63. Of the seven total hCoVs, only SARS-CoV-2, SARS-CoV, and MERS-CoV cause severe disease in humans while the remaining cause symptoms of the common cold. The hCoV with the greatest similarity to the SARS-CoV-2 genome is SARS-CoV, with the next most similar being MERS-CoV. However, all of these viral genomes are less similar to SARS-CoV-2 when compared to the previously mentioned bat CoVs.

For comparison with other variants of SARS-CoV-2 that have emerged since Wuhan-Hu-1, there are two variants currently designated as "variants of concern" by the CDC: Omicron B.1.529 and Delta B.1.617.2 (CDC, 2022). The omicron variant is considered of concern due to its increased transmissibility, though the severity of illness may be less than with prior variants. The Delta variant is of concern due to both increased transmissibility and

potentially increased severity of illness. Therefore, these strains are also strong candidates for our comparative analyses. All other variants listed by the CDC (Alpha, Beta, Gamma, Epsilon, Eta, Iota, Kappa, B.1.617.3, Zeta, and Mu) are currently designated "variants being monitored."

Molecular and Genomic Structure of Sars-CoV-2

For the topic of SARS-CoV-2 genomic characteristics, the primary goals were to learn the basics of the genomic structure in order to better understand how to approach the genomic and intragenomic analyses. The SARS-CoV-2 genome is ~30kb and is single-stranded RNA (Kadam, et al., 2021). The genome itself is structured like an mRNA molecule, including the 5' cap and 3' poly-A tail. The Wuhan-Hu-1 genome contains just 12 protein coding genes or ORFs. These include, in 5' to 3' order, ORF1a, ORF1ab, S(spike), ORF3a, E(envelope), M(membrane), ORF6, ORF7a, ORF7b, ORF8, N(nucleocapsid)/ORF9, and ORF10. The coding sequence of the first ORF, ORF1a, also makes up a large portion of the second ORF, ORF1ab, due to a ribosomal -1 frameshift during translation at the last position in ORF1a which is likely caused by a pseudoknot found in the secondary structure of the RNA genome. This is an important feature to note because each ORF1a and ORF1ab are listed independently as features in the GenBank file of the Wuhan-Hu-1 genome used for our analysis. Additionally, there is a small region of overlap between ORF7a and ORF7b, also the result of a frameshift. These changes in frame are important to consider when determining the codons from the genomic sequence. A graphical representation of the complete genome is presented in Figure 1 below. While ORF9b is represented as a separate gene in this graphic, ORF9 and N are listed as the same CDS feature in the reference sequence retrieved from NCBI's GenBank. The arrangement of ORFs in the genome significantly resembles the SARS-CoV and MERS-CoV (Naqvi et al., 2020)

Figure 1

SARS-CoV-2 Genome

Note: Graphical representation of SARS-CoV-2 genome. Adapted from "SARS-CoV-2, the pandemic coronavirus: Molecular and structural insights," by S.B. Kadam, G.S. Sukhramani, P. Bishnoi, A.A. Pable, & V.T. Barvkar, 2021, *Journal of Basic Microbiology, 63*(3), 180-202

Codon Usage Bias

Alternative codons of an amino acid are not used randomly in an organism. This non-random selection of synonymous codons results in codon usage bias (CUB) in species (Sharp & Li, 1987). CUB is an important measure of genome evolution. The bias may be analyzed on the protein level where preferential amino acid usage is considered or synonymous codon usage where preferences of codon usage for the same amino acid are measured. Especially for the viral species, the preference of codon depends on the host. The reasons such as available tRNAs in the host or specific patterns triggering defense mechanisms create selective pressure on the virus (Dilucca et al., 2020).

There are several different calculations and indexes that can be employed to quantify and evaluate this characteristic of a gene or genome. Sharp & Li (1987) described a method for calculating the relative synonymous codon usage (RSCU) for each codon and the codon adaptive index (CAI) for each gene. There are varying numbers of synonymous codons per amino acid,

ranging from no redundancy in codons for methionine and tryptophan (each has only one possible codon) to six synonymous codons for leucine. This means that the expected frequency assuming equal usage also varies depending on the amino acid. Due to this variation, calculating the simple frequencies for each codon out of all 64 possible does not accurately reflect codon usage bias. The RSCU value is calculated as the frequency of a given codon divided by the expected frequency when compared to only other synonymous codons for the same amino acid. By including the expected frequency for each amino acid in the calculation, this metric better reflects actual codon bias. When analyzing RSCU data generated for a codon, an RSCU value of more than 1.0 has a positive codon usage bias, while a value of less than 1.0 has a negative codon usage bias. When the RSCU value is equal to 1.0, this indicates the codon is chosen equally and randomly.

The CAI value for a gene is calculated from RSCU values and reflects how well adapted a gene's codons are to the species overall. It assumes that highly expressed genes in a species are the best adapted to that species and thus uses the RSCU values from those highly expressed genes as a reference set to normalize the other genes against. In the case of viral genes, the reference RSCU values are to be derived from the host genome's most highly expressed genes rather than the virus's own and thus reflect how well adapted the virus is to its host. Since CAI calculations involve data from the host, it may be useful to compare how well Wuhan-Hu-1 is adapted to the human host as compared to other human SARS-CoV-2 strains, but it may not be indicated for comparison against non-human pathogenic strains.

Another metric that appears often in the literature for codon usage bias is the effective number of codons or ENC. ENC is used to evaluate the codon usage bias of SARS-CoV-2 in

multiple publications (Hou, 2020; Dilucca, et al., 2020; Tort, et al., 2020). Wright (1990) first described ENC as a metric to express the degree with which a gene's codon usage deviates from equal usage with scores from 20 (highly biased) to 61 (no bias). However, weaknesses with this metric have been identified and several alternative methods for calculation have been proposed since its first introduction in 1990 (Fuglsang, 2004; Banerjee, et al., 2005; Fuglsang, 2006; Sun, Yang, & Xia, 2013).

Overall, it seems that RSCU fulfills the needs of this project, and it has also been used in many examples of SARS-CoV-2 codon usage analysis that have been previously published, including those that also used ENC (Hou, 2020; Dilucca, et al., 2020; Tort, et al., 2020). After discussing all of this information during the development of our python script, it was decided that we would proceed with calculations for RSCU values only, but that CAI values or ENC values could be considered as additions for a more thorough analysis later.

Methods

Calculating RSCU with Python

Programming Design Decisions

Our objective was to create a code to find codon frequencies for the Wuhan-1 strain. Our approach is to create a universal python script for any strain, so the script can be run multiple times for any strain and generate data as much as needed. Since the coding regions of the Wuhan-1 strain and other strains are already defined in the GenBank database, we used GenBank files as our initial input. We used Biopython libraries (Biopython, 2021) to parse GenBank files and extract CDS sequences. Raw codon frequencies and RSCU are calculated by our own functions by using the formulas defined by Sharp and Li (1987) in our script.

Once the base for our python script to calculate codon frequencies and RSCU values for the Wuhan-Hu-1 strain overall was completed, a simple test was devised to verify that the calculations were being performed correctly. The quality and utility of the final data analysis is dependent on having quality data to analyze, so it is integral to ensure that our calculations are correct and the python script is producing output as intended. The expected frequencies assuming equal usage for each codon based on the total synonymous codons for its associated amino acid were calculated in Excel. Since the python script accepts GenBank formatted sequence files and extracts the CDS, a test GenBank file containing a single CDS with equal codon usage for all codons was then created. Thus, if the python script calculates the frequencies and RSCU values correctly, the output from processing this test GenBank file with our python script should match the expected values calculated in Excel exactly. Additionally, the RSCU values calculated by the

python script should be equal to 1.0 for all codons. The Excel workbook was used to compare the script output against the expected frequencies with vlookup formulas.

This test sequence was processed with the script to identify a few bugs in the functions and calculations that were causing 'divide by zero' errors and some incorrect RSCU values. These errors in the script were resolved and the expected output for the test sequence was ultimately produced for both the frequencies and RSCU values. Screenshots from the analysis of this final output are included in Figures 2 and 3 below.

Figure 2

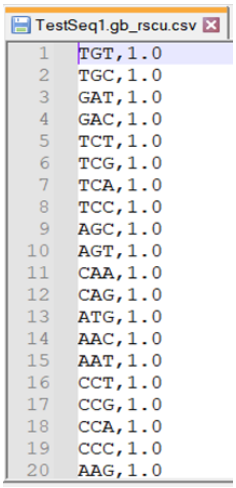
Test of Frequency Calculations

AA	Codon	Script Output	Expected	Match	Total Mismatch
F	TTT	0.5	0.5	Yes	0
F	TTC	0.5	0.5	Yes	
L	TTA	0.167	0.167	Yes	
L	TTG	0.167	0.167	Yes	
L	CTT	0.167	0.167	Yes	
L	CTC	0.167	0.167	Yes	
L	CTA	0.167	0.167	Yes	
L	CTG	0.167	0.167	Yes	
I	ATT	0.333	0.333	Yes	
I	ATC	0.333	0.333	Yes	
I	ATA	0.333	0.333	Yes	
M	ATG	1	1	Yes	
V	GTT	0.25	0.25	Yes	
V	GTC	0.25	0.25	Yes	
V	GTA	0.25	0.25	Yes	
V	GTG	0.25	0.25	Yes	
Y	TAT	0.5	0.5	Yes	
Y	TAC	0.5	0.5	Yes	
-	TAA	0.333	0.333	Yes	
-	TAG	0.333	0.333	Yes	
H	CAT	0.5	0.5	Yes	
H	CAC	0.5	0.5	Yes	
Q	CAA	0.5	0.5	Yes	
Q	CAG	0.5	0.5	Yes	
N	AAT	0.5	0.5	Yes	

Note: Excel workbook comparison between the script output and the expected output values for the test sequence showing that no mismatches between the values were identified.

Figure 3

Test of RSCU Calculations



Note: Python script RSCU output for the test sequence with all values at 1.0, as expected.

After the calculations of the python script were verified as working correctly, further modifications were made to the script to produce the final output. In addition to calculating the RSCUs for the genome overall, the script also required functions to allow intragenomic comparisons. The most obvious way to divide up the genome for intragenomic comparison would be by gene or CDS, and the desire was for the script to do that automatically from the input GenBank file. The initial intention was also to incorporate the ability to analyze by windows of arbitrary size as it was requested in both the original project description and kickoff meeting. Additionally, it was requested by a group member that we add an option to convert the codons to a numerical value as that may facilitate further downstream analyses. After adding the modifications and additional functions, a script was produced that can output RSCUs for the

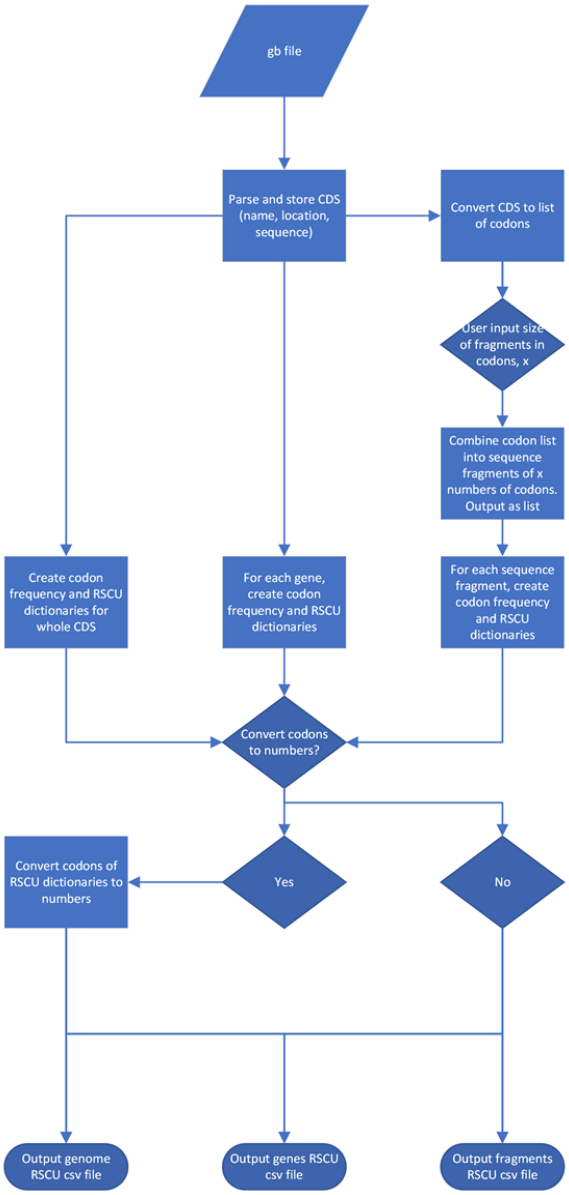
whole genome, for each gene, and for each fragment of arbitrary size. All three methods were performed for each input GenBank file, according to the flowchart in Figure 4 below.

Assumptions and Constraints

Dividing the genome into fragments of arbitrary size was complicated by the need to keep codons intact despite differing frames between genes, overlapping regions, and the frameshift encountered within ORF1ab. The initial strategy was to extract each CDS sequence based on the loci provided in the Genbank file, concatenate them, and then construct the sequence as a list of codons rather than nucleotides. Then, the sequence can be divided into any user-input number of whole codons. However, upon further review, it was noted that this method did not account for the overlapping sequences.

Figure 4

Flow Chart of Python Script



Note: Flowchart depicts the python script to calculate RSCU for the whole genome, for each gene, and for fragments of arbitrary size for each input GenBank file. Also includes an option to convert codons to a numeric value.

ORF1ab contains ORF1a in its entirety. Since the python script extracts and uses all CDS features, and since ORF1a and ORF1ab are each listed as separate CDS in the GenBank file for Wuhan-Hu-1, that means the 4401 codons of ORF1a (more than a third of the entire genome) are being duplicated in the calculations. This impacts the RSCU calculations for both the whole genome and the fragments of arbitrary size. To rectify this, the function that parses the GenBank file and extracts the CDS was modified with "if" statements in order to exclude CDS that are contained entirely within another CDS. An additional small overlap is present between the end of ORF7a and the beginning of ORF7b. However, ORF7b is translated in a different frame from ORF7a so the codons differ and are not being duplicated. After testing with other GenBank files for other viruses, it was also found that not all are complete with gene names. Thus, the function was modified again to extract the protein product name in the absence of a gene name.

After the discussion during the status update presentation, the right branch of the flow chart depicted in Figure 4 was commented out and output for fragments of arbitrary size is no longer being produced. The final python script used for our analyses consists of the left and center branches to output whole genome RSCU and RSCU for each gene in two separate csv files.

Visualizing RSCU Data With R

Once the python code for generating the codon frequencies and RSCU values for the Wuhan-Hu-1 strain was complete, different analysis approaches were constructed in R for visualizing the data. The first program created was for visualizing the codon frequencies of the Wuhan-Hu-1 strain as well as the other pathogenic and non-pathogenic strains. The ggplot2 package of R was used for creating the graphics which allowed for data variables to be mapped

to aesthetics and further add layers and faceting specifications depending on the end goal. First, the .csv file is read into R followed by the ggplot() function being called for that dataset. Next, the aesthetics of the graphical elements were declared with the x-axis representing the codon, the y axis the codon frequency, and the graphs were filled for the AA that each codon represents as seen in Figures 10-14. This code provided a working method for identifying the codon patterns amongst the genomes of the different viral strains. A slightly altered code for visualizing the RSCU values across genes and gene fragments of the different viral strains was also created using the ggplot2 package in R. For this way of analyzing the data, the code's facet wrap was changed to output long bar plots of RSCU values for the multiple viral strains in one graph for either a specific gene or gene fragment as seen in Figures 5 and 6.

Results

Codon Usage Analysis in Whole Genome

To begin our analysis, codon frequencies and RSCU values were calculated for the Wuhan-Hu-1 genome as well as several other viruses for comparison. These results were analyzed in several different ways. When SARS-CoV-2 (Wuhan-Hu-1 strain) is compared to SARS, MERS, pangolin, and RaTG13 (bat) coronaviruses, similar RSCU distribution is observed among species. Figure 5 shows the heatmap of this comparison. Also, Figure 6 shows the RSCU values of each virus grouped by codon. H-Cov-EMC shows a little divergence from other strains of the virus. These data are concordant with the study of Naqvi and colleagues (2020) which shows that MERS-CoV has a little divergence when it is compared with the SARS-CoV-2, BATCoV HKU3, and SARS-CoV.

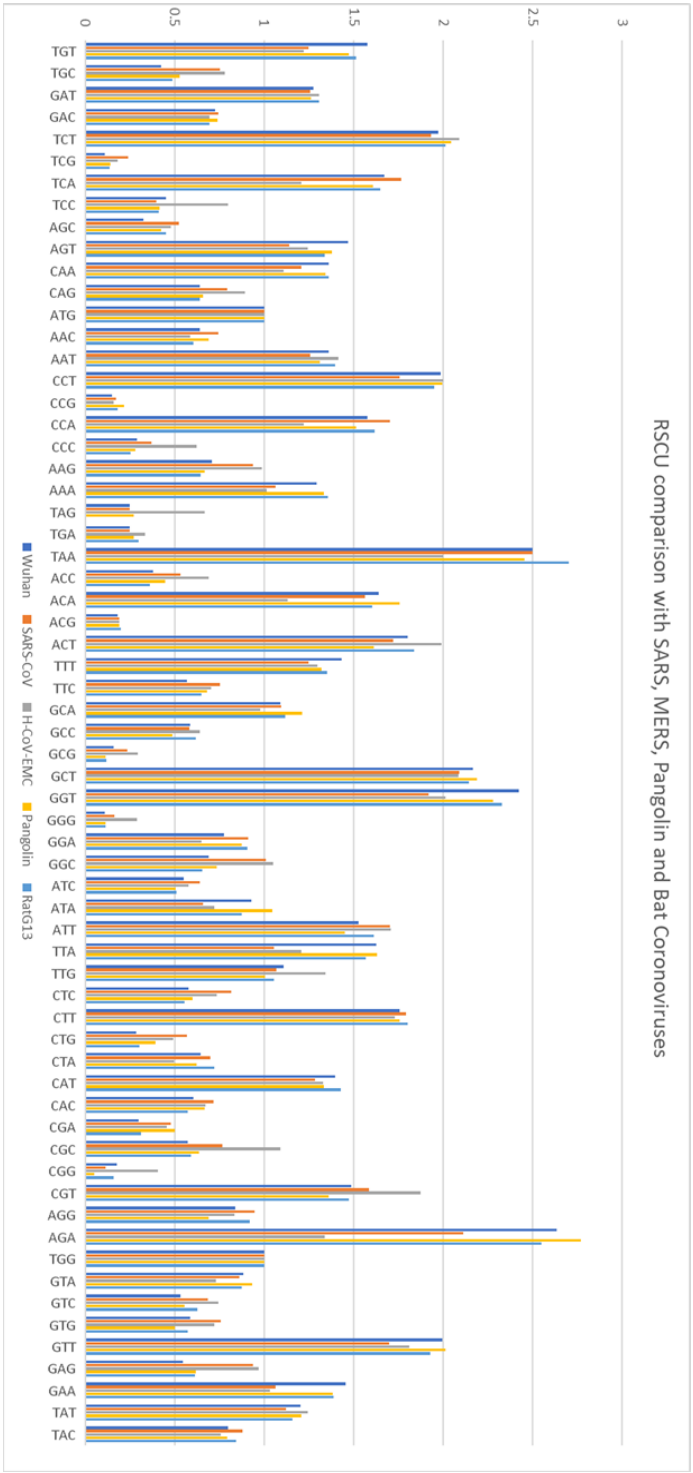
Figure 5

Comparison of SARS, MERS, Pangolin, and Bat coronaviruses RSCU values.

	Wuhan	SARS-CoV	H-CoV-EMC	Pangolin	RatG13
TGT	1.576	1.249	1.219	1.471	1.512
TGC	0.424	0.751	0.781	0.529	0.488
GAT	1.274	1.256	1.304	1.263	1.306
GAC	0.726	0.744	0.696	0.737	0.694
TCT	1.973	1.933	2.091	2.046	2.015
TCG	0.107	0.239	0.18	0.138	0.137
TCA	1.671	1.767	1.206	1.606	1.649
TCC	0.452	0.396	0.799	0.413	0.412
AGC	0.327	0.525	0.478	0.422	0.449
AGT	1.47	1.141	1.245	1.376	1.337
CAA	1.36	1.209	1.11	1.343	1.359
CAG	0.64	0.791	0.89	0.657	0.641
ATG	1	1	1	1	1
AAC	0.641	0.742	0.588	0.689	0.603
AAT	1.359	1.258	1.412	1.311	1.397
CCT	1.986	1.754	2	1.995	1.949
CCG	0.151	0.17	0.16	0.216	0.182
CCA	1.577	1.704	1.221	1.512	1.616
CCC	0.287	0.371	0.62	0.278	0.253
AAG	0.709	0.935	0.988	0.669	0.643
AAA	1.291	1.065	1.012	1.331	1.357
TAG	0.25	0.25	0.667	0.273	0
TGA	0.25	0.25	0.333	0.273	0.3
TAA	2.5	2.5	2	2.455	2.7
ACC	0.38	0.53	0.689	0.447	0.361
ACA	1.641	1.562	1.129	1.754	1.603
ACG	0.181	0.19	0.19	0.188	0.2
ACT	1.799	1.718	1.991	1.611	1.836
TTT	1.432	1.247	1.299	1.318	1.351
TTC	0.568	0.753	0.701	0.682	0.649
GCA	1.088	1.096	0.977	1.211	1.119
GCC	0.587	0.58	0.642	0.488	0.618
GCG	0.157	0.235	0.295	0.114	0.117
GCT	2.168	2.089	2.085	2.187	2.145
GGT	2.424	1.918	2.011	2.278	2.328
GGG	0.11	0.164	0.287	0.111	0.111
GGA	0.776	0.911	0.651	0.875	0.906
GGC	0.69	1.007	1.051	0.736	0.655
ATC	0.548	0.64	0.576	0.505	0.511
ATA	0.927	0.658	0.719	1.046	0.876
ATT	1.525	1.703	1.705	1.45	1.613
TTA	1.626	1.056	1.205	1.63	1.568
TTG	1.108	1.069	1.344	1.004	1.052
CTC	0.576	0.814	0.735	0.6	0.555
CTT	1.758	1.794	1.728	1.754	1.8
CTG	0.286	0.566	0.49	0.391	0.303
CTA	0.646	0.7	0.497	0.62	0.723
CAT	1.394	1.282	1.327	1.333	1.429
CAC	0.606	0.718	0.673	0.667	0.571
CGA	0.299	0.48	0.457	0.499	0.312
CGC	0.574	0.768	1.092	0.636	0.59
CGG	0.175	0.112	0.408	0.052	0.156
CGT	1.484	1.584	1.875	1.358	1.474
AGG	0.836	0.944	0.832	0.688	0.919
AGA	2.632	2.112	1.337	2.768	2.549
TGG	1	1	1	1	1
GTA	0.885	0.862	0.728	0.932	0.875
GTC	0.531	0.684	0.742	0.555	0.626
GTG	0.587	0.757	0.723	0.499	0.574
GTT	1.997	1.697	1.808	2.013	1.925
GAG	0.546	0.939	0.968	0.616	0.612
GAA	1.454	1.061	1.032	1.384	1.388
TAT	1.201	1.122	1.244	1.205	1.159
TAC	0.799	0.878	0.756	0.795	0.841

Figure 6

RSCU comparison of SARS, MERS, Pangolin, and Bat coronaviruses.



When comparing codon usage by the whole genome among different host species as depicted in Figures 7 and 8, there's an observable difference in codon frequencies by species.

Figure 7

Codon Frequencies Within Different Hosts.

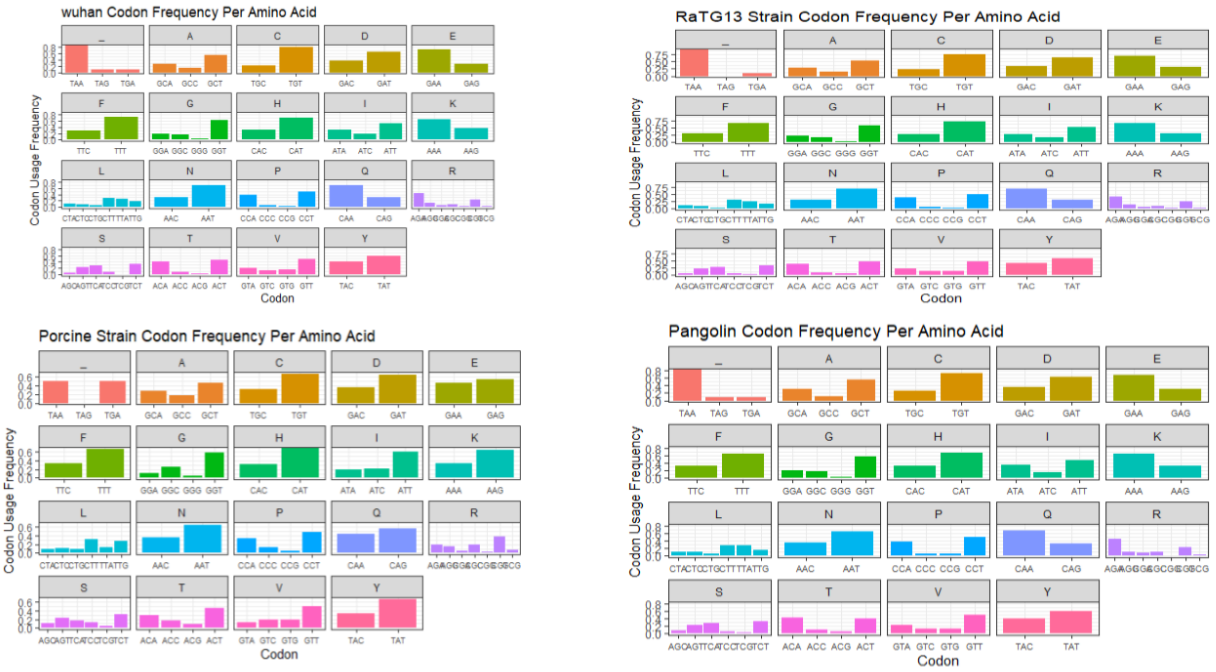
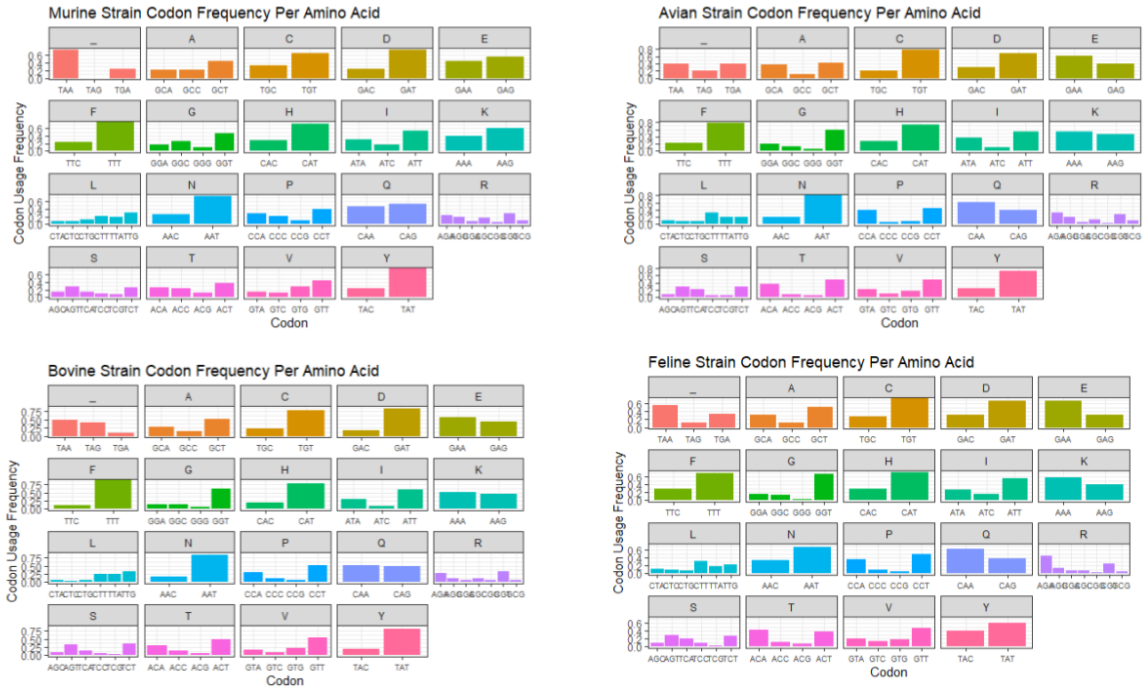


Figure 8

Additional Codon Frequencies Within Different Hosts



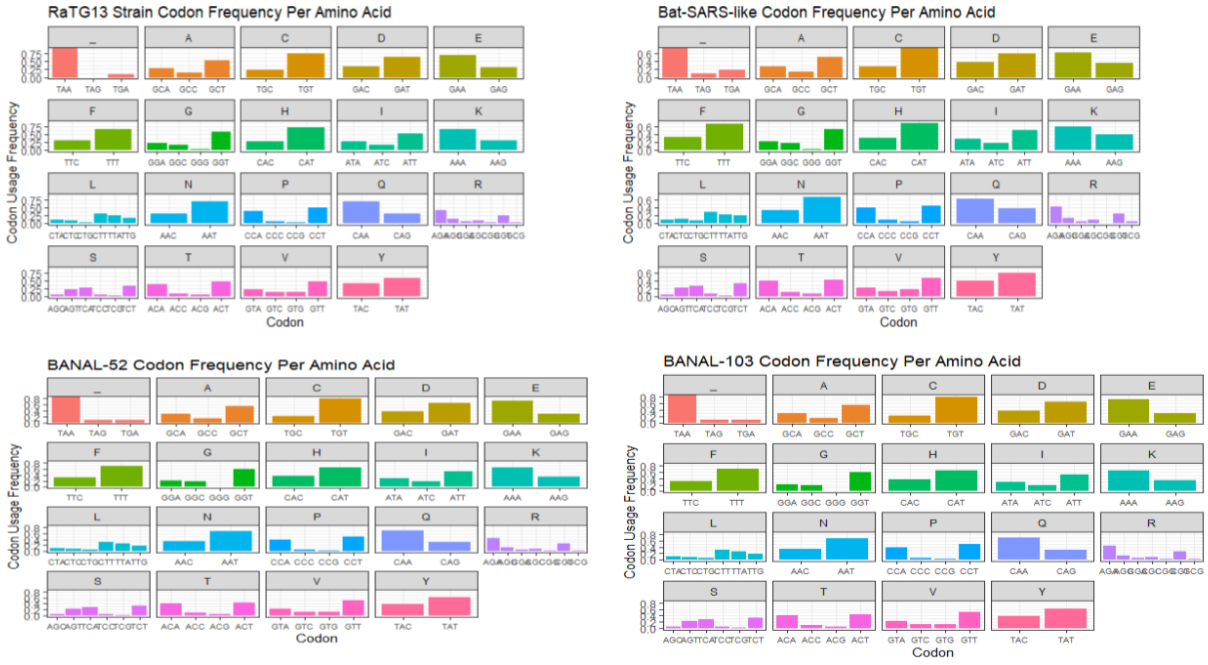
However, when observing codon frequencies among different strains of SARS-CoV-2 isolated from the same host, the differences among frequencies are far less observable and a bit more subtle as seen in Figures 9 and 10.

Figure 9

Codon frequencies within the same host (human).

Figure 10

Codon frequencies within the same host (bat).



Nonetheless, when we compare codon frequencies of the historic strain isolated from a bat host (RaTG13) to those recently found in Laos to the Wuhan-Hu-1 strain, the observable difference becomes less prominent suggesting a natural jump in host species.

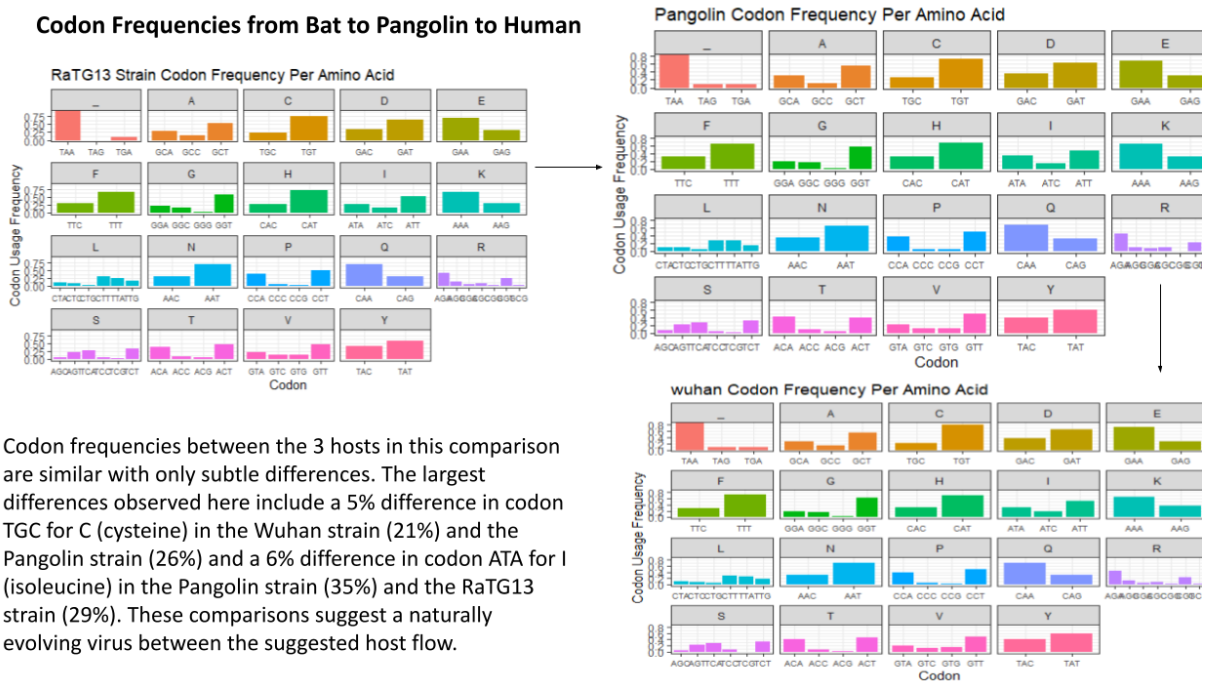
Figure 11

BANAL-20-52, BANAL-20-103, RaGT13, and Wuhan-Hu-1 codon frequencies comparison.

The results reported thus far include codon frequencies calculated from the whole genome of coronaviruses isolated from different hosts while comparing them from various different host species, the same species, and from different species suspected to have transmitted the virus to one another. Lastly, Figure 12 shows these same codon frequencies previously described but these comparisons show the RaTG13 strain, a pangolin strain, and the Wuhan-Hu-1 strain demonstrate vast similarity among codon usage of the different hosts considered likely responsible for transmitting the virus to humans.

Figure 12

Codon frequencies among host species flow from animal (non-pathogenic) to human (pathogenic) host.



Note: Subtle codon usage differences were observed in 3 different hosts showing the suggested host flow of the coronavirus from the historical bat host to the first identified human host strain responsible for the pandemic caused by SARS-CoV-2 infection.

Codon Usage Analysis By Gene

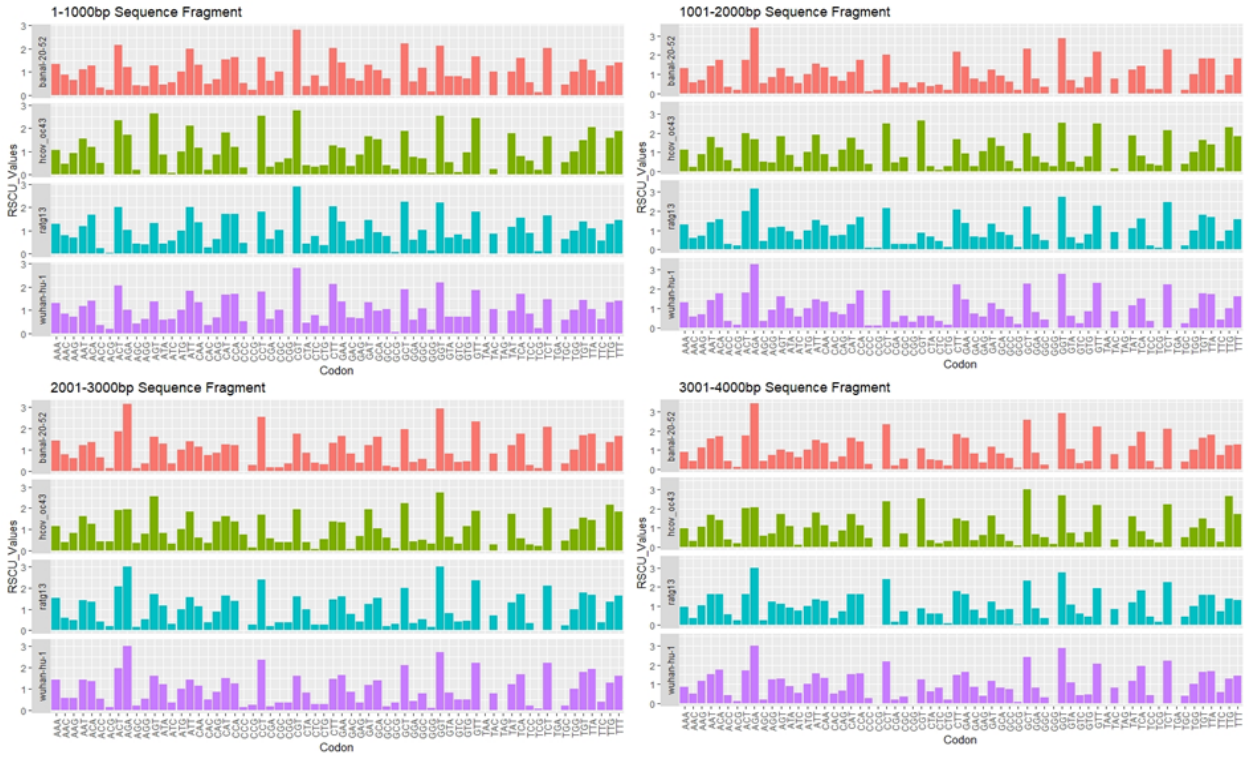
In this study, RSCU values among different genes of coronaviruses compared to the Wuhan-Hu-1 strain were analyzed as seen in Figure 13. It was observed that for the spike (S) protein gene the overrepresented codons included ACT, CTT, GCT, GTT, and TCT for all the strains of coronaviruses. Also observed in the membrane protein gene in Figure 13, the Wuhan-Hu-1 strain did not use TGC for Cysteine unlike the other strains. In the envelope protein gene, codons CAA, CAC, CAG, CAT, CCA, CCC, and CCG were found not to be used in the BANAL-20-52, RaTG13, or Wuhan-Hu-1 strains which differed from the human coronavirus, HCoV-OC43, in which CCA, CAT, and CAA were actually represented. The codon ACT for Threonine was found in the envelope protein to be overrepresented in the human coronavirus, HCoV-OC43, when compared to the Wuhan-Hu-1 strain and other non-human coronaviruses. These findings remain consistent with other recent reports including Wei Hou's, *Characterization of codon usage pattern in SARS-CoV-2*, in 2020.

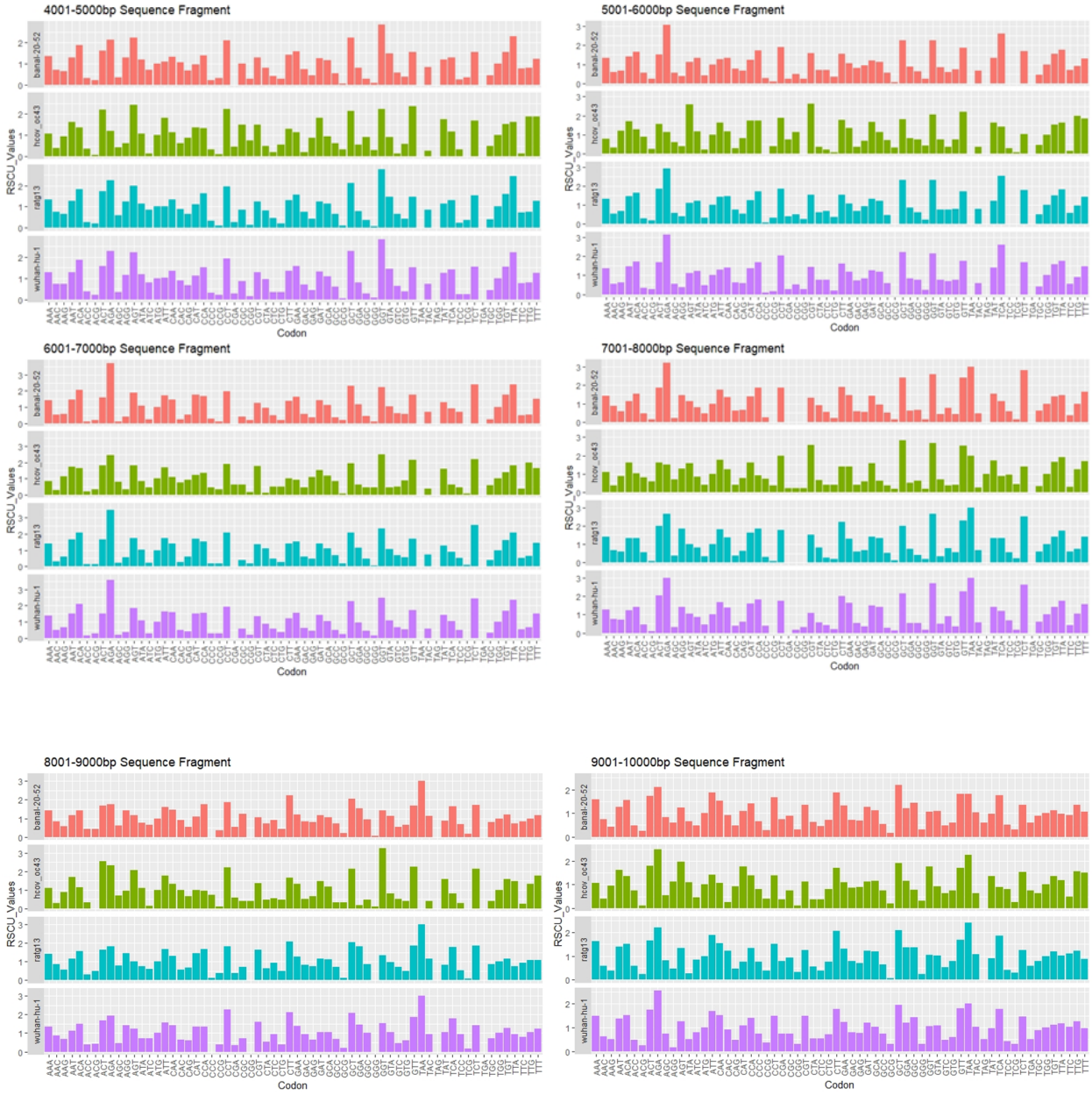
Figure 13

RSCU values for different genes of SARS-CoV-2 viral strains

Figure 14

RSCU values for different gene fragments of SARS-CoV-2 viral strains





Codon usage of Wuhan-Hu-1 strain was further compared gene by gene. Figure 15 shows the relative codon usage of each gene. Increased usage of TCA, AGG, AGA codons are observed in ORF7b, ORF6, and ORF7a respectively. These ORFs encode accessory proteins (Kadam et.al., 2021).

Figure 15

Heatmap of Wuhan-Hu-1 strain RSCU values by genes

	TGT	TGC	GAT	GAC	TCT	TCG	TCA	TCC	AGC	AGT	CAA	CAG	ATG	AAC	AAT	CCT
ORF1ab	1.63	0.37	1.29	0.71	2.03	0.07	1.68	0.36	0.30	1.57	1.36	0.64	1.00	0.60	1.40	1.99
S	1.40	0.60	1.39	0.61	2.24	0.12	1.58	0.73	0.30	1.03	1.48	0.52	1.00	0.77	1.23	2.00
ORF3a	0.86	1.14	1.08	0.92	0.82	0.00	2.18	1.09	0.55	1.36	1.11	0.89	1.00	1.00	1.00	2.33
E	0.67	1.33	2.00	0.00	3.00	0.75	0.75	0.00	0.75	0.75	0.00	0.00	1.00	0.40	1.60	4.00
M	2.00	0.00	0.33	1.67	0.80	0.40	1.20	1.20	0.80	1.60	1.00	1.00	1.00	1.27	0.73	0.80
ORF6	0.00	0.00	1.50	0.50	3.00	0.00	1.50	1.50	0.00	0.00	1.33	0.67	1.00	0.50	1.50	0.00
ORF7a	1.00	1.00	1.00	1.00	2.57	0.00	2.57	0.00	0.86	0.00	1.60	0.40	1.00	1.00	1.00	2.67
ORF7b	1.00	1.00	1.00	1.00	0.00	0.00	6.00	0.00	0.00	0.00	2.00	0.00	1.00	0.00	2.00	0.00
ORF8	1.43	0.57	1.14	0.86	1.33	0.67	2.00	0.67	0.00	1.33	1.00	1.00	1.00	0.00	2.00	2.29
N	0.00	0.00	1.17	0.83	1.30	0.32	1.46	0.49	0.97	1.46	1.54	0.46	1.00	0.55	1.45	1.14
ORF10	0.00	2.00	2.00	0.00	3.00	0.00	0.00	0.00	0.00	3.00	2.00	0.00	1.00	1.20	0.80	0.00

	CCG	CCA	CCC	AAG	AAA	TAG	TGA	TAA	ACC	ACA	ACG	ACT	TTT	TTC	GCA	GCC
ORF1ab	0.15	1.62	0.25	0.71	1.29	0.00	0.00	3.00	0.36	1.70	0.18	1.76	1.46	0.54	1.08	0.58
S	0.00	1.72	0.28	0.75	1.25	0.00	0.00	3.00	0.41	1.65	0.12	1.81	1.53	0.47	1.37	0.41
ORF3a	0.67	1.00	0.00	0.73	1.27	0.00	0.00	3.00	0.33	1.00	0.50	2.17	1.14	0.86	0.92	0.92
E	0.00	0.00	0.00	0.00	2.00	0.00	0.00	3.00	0.00	2.00	1.00	1.00	0.80	1.20	0.00	1.00
M	0.80	2.40	0.00	0.86	1.14	0.00	0.00	3.00	0.92	0.92	0.62	1.54	0.91	1.09	0.84	0.42
ORF6	0.00	4.00	0.00	0.50	1.50	0.00	0.00	3.00	0.00	0.00	0.00	4.00	2.00	0.00	4.00	0.00
ORF7a	0.00	1.33	0.00	0.29	1.71	0.00	3.00	0.00	0.00	2.80	0.00	1.20	1.40	0.60	1.33	0.44
ORF7b	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	4.00	1.00	1.00	0.00	4.00
ORF8	0.57	0.57	0.57	0.00	2.00	0.00	0.00	3.00	0.00	2.40	0.00	1.60	1.00	1.00	1.60	0.00
N	0.29	1.57	1.00	0.65	1.35	0.00	0.00	3.00	0.75	1.00	0.25	2.00	0.46	1.54	0.86	0.76
ORF10	4.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	2.00	2.00	0.00	1.50	0.50	2.00	0.00

	GCG	GCT	GGT	GGG	GGA	GGC	ATC	ATA	ATT	TTA	TTG	CTC	CTT	CTG	CTA	CAT
ORF1ab	0.13	2.21	2.55	0.10	0.74	0.61	0.50	1.02	1.48	1.81	1.04	0.55	1.66	0.25	0.69	1.41
S	0.10	2.13	2.29	0.15	0.83	0.73	0.55	0.71	1.74	1.56	1.11	0.67	2.00	0.17	0.50	1.53
ORF3a	0.00	2.15	2.00	0.00	1.14	0.86	0.71	1.00	1.29	0.60	1.80	1.00	2.00	0.40	0.20	1.00
E	2.00	1.00	4.00	0.00	0.00	0.00	1.00	1.00	1.00	0.43	0.86	0.00	3.00	0.86	0.86	0.00
M	0.21	2.53	1.43	0.00	1.71	0.86	0.90	0.45	1.65	0.69	0.69	1.03	2.06	0.69	0.86	1.60
ORF6	0.00	0.00	0.00	0.00	0.00	0.00	0.30	1.20	1.50	2.25	0.00	1.50	0.75	0.00	1.50	2.00
ORF7a	0.44	1.78	1.00	0.00	1.00	2.00	0.38	1.13	1.50	0.80	0.40	0.80	2.40	1.20	0.40	0.67
ORF7b	0.00	0.00	0.00	0.00	0.00	0.00	0.60	0.00	2.40	1.64	0.55	0.00	2.18	1.09	0.55	1.00
ORF8	0.00	2.40	2.40	0.00	1.60	0.00	1.50	0.00	1.50	3.60	1.20	0.00	1.20	0.00	0.00	1.00
N	0.32	2.05	0.93	0.37	1.21	1.49	0.86	0.21	1.93	0.44	2.00	0.44	1.78	0.67	0.67	1.50
ORF10	0.00	2.00	0.00	0.00	0.00	4.00	0.00	3.00	0.00	0.00	1.50	3.00	0.00	0.00	1.50	0.00

	CAC	CGA	CGC	CGG	CGT	AGG	AGA	TGG	GTA	GTC	GTG	GTT	GAG	GAA	TAT	TAC
ORF1ab	0.59	0.25	0.61	0.17	1.45	0.76	2.75	1.00	0.92	0.54	0.59	1.95	0.54	1.46	1.24	0.76
S	0.47	0.00	0.14	0.29	1.29	1.43	2.86	1.00	0.62	0.87	0.54	1.98	0.58	1.42	1.48	0.52
ORF3a	1.00	0.00	1.00	0.00	1.00	1.00	3.00	1.00	1.12	0.48	0.16	2.24	0.18	1.82	0.94	1.06
E	0.00	2.00	0.00	0.00	2.00	0.00	2.00	0.00	0.92	0.31	0.62	2.15	1.00	1.00	0.00	2.00
M	0.40	0.43	0.86	0.00	2.14	1.29	1.29	1.00	2.00	0.00	1.00	1.00	0.29	1.71	0.89	1.11
ORF6	0.00	0.00	0.00	0.00	0.00	6.00	0.00	1.00	0.00	0.00	0.00	4.00	1.60	0.40	1.00	1.00
ORF7a	1.33	0.00	0.00	0.00	1.20	0.00	4.80	0.00	1.00	0.50	0.50	2.00	1.00	1.00	0.80	1.20
ORF7b	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	4.00	0.00	2.00	2.00	0.00
ORF8	1.00	0.00	0.00	0.00	3.00	0.00	3.00	1.00	1.33	0.00	0.67	2.00	0.67	1.33	1.71	0.29
N	0.50	1.03	1.03	0.41	1.24	0.21	2.07	1.00	0.50	1.50	1.00	1.00	0.67	1.33	0.36	1.64
ORF10	0.00	0.00	0.00	0.00	3.00	0.00	3.00	0.00	2.00	0.00	0.00	2.00	0.00	0.00	1.33	0.67

Codon Usage Analysis in ORF1ab Gene

Additional analysis was conducted to find out if there was any correlation of codon bias observed in just the RSCU values of ORF1ab in particular. The expectation from prior research is that data should show preferential use of A-ended and/or U-ended codons as one trend of codon bias (Hou, 2020). Another trend of note is G and C-ended codons should be significantly lower than A and U-ended codons of trends to observe.

The RSCU gene frequencies of the optimal codons within ORF1ab were compared separately for each of the initial strains and highlighted based on its frequency of codon seen in Figure 16. Codons highlighted in red are abundant codons or overly expressed (RSCU values >1.5) and codons highlighted in yellow are considered optimal codons (RSCU values 1-<1.5). All codon sets that did not have this trend were removed.

Figure 16

The RSCU gene frequencies of the optimal codons within ORF1

Wuhan-1	ORF1ab
GAT	1.290488
CAA	1.364017
ATG	1
AAT	1.395833
AAA	1.294931
TTT	1.455587
GCA	1.075975
ATA	1.023324
ATT	1.478134
TTG	1.041916
CAT	1.406897
CGT	1.45082
TGG	1
GAA	1.458824
TAT	1.241791

Banal-20-52	ORF1ab polyprotein
GAT	1.266666667
CAA	1.327731092
ATG	1
AAT	1.336814621
AAA	1.321016166
TTT	1.435158501
GCA	1.125
ATA	1.020172911
ATT	1.5129683
TTG	1.029850746
CAT	1.365517241
CGT	1.5
TGG	1
GAA	1.452380952
TAT	1.220238095

HCov-OC-43	Pp1ab
GAT	1.70989
CAA	1.025862
ATG	1
AAT	1.677966
AAA	1.006623
TTT	1.799499
GCA	1.102564
ATA	0.959016
ATT	1.811475
TTG	2.104135
CAT	1.615385
CGT	2.174672
TGG	1
GAA	1.162791
TAT	1.697917

ratg13	orf1ab
GAT	1.317949
CAA	1.319328
ATG	1
AAT	1.431525
AAA	1.325635
TTT	1.427746
GCA	1.102296
ATA	0.982759
ATT	1.534483
TTG	1.065672
CAT	1.472222
CGT	1.469388
TGG	1
GAA	1.424332
TAT	1.190476

- Codons highlighted in red are abundant codons or overly expressed (RSCU values >1.5).
- Codons highlighted in yellow are considered optimal codons (RSCU values 1-<1.5).

Note: HCov-OC-43 is the outlier strain that has the most abundant/over-expressed codons compared to Wuhan-1, RaTG13, and Banal-20-52. A- and T-ending codons are predominant in the HCov-OC-43 strain with higher frequencies from the overexpressed codons observed.

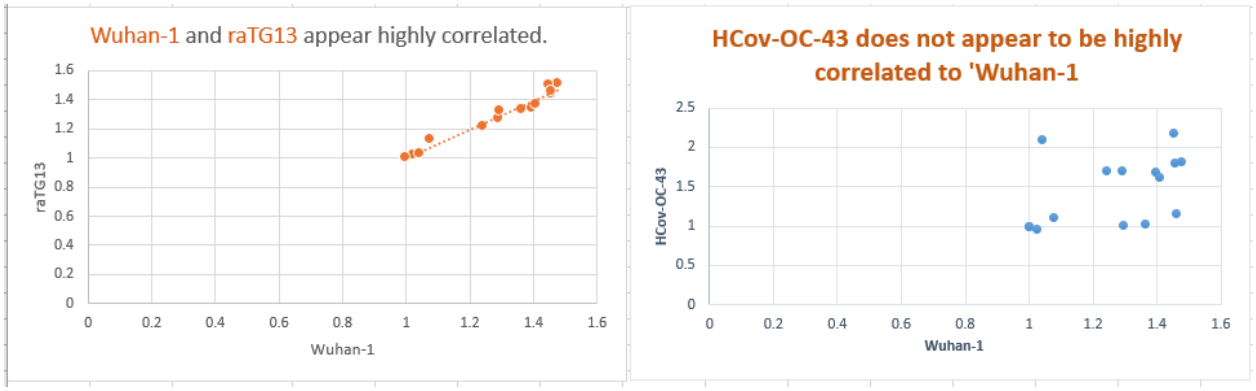
It is observed in the Wuhan-1 reference (SARS-CoV-2) the predominant pattern of A- and T rich codons are seen in the optimal codon frequencies and G and C-ended codons are less as expected. Most of the GC ended codon values were not optimal and will have a negative impact on codon usage bias. (RSCU values <1 are not optimal). HCov-OC-43 is the outlier strain, as seen in Figure 16, that has the most abundant/over-expressed codons compared to

Wuhan-1, RaTG13, and Banal-20-52. A- and T-ending codons are predominant in the HCov-OC-43 strain with higher frequencies from the overexpressed codons observed.

Mutational pressure is considered a significant driver of codon usage patterns in SARS-CoV-2. The GC content analysis was measured within ORF1, as this compositional constraint is a major influence on synonymous codon usage. The GC analysis in codon usage bias is used to show the correlation with the outlier strain HCoV-43 in Figure 17. RSCU frequencies within the ORF1 gene were plotted for comparison which showed Wuhan-Hu-1 and RaTG13 are highly correlated whereas HCoV-43 appears less correlated to Wuhan-1.

Figure 17

Correlation Analysis of Wuhan-1, RaTG13, and HCoV-oc-43

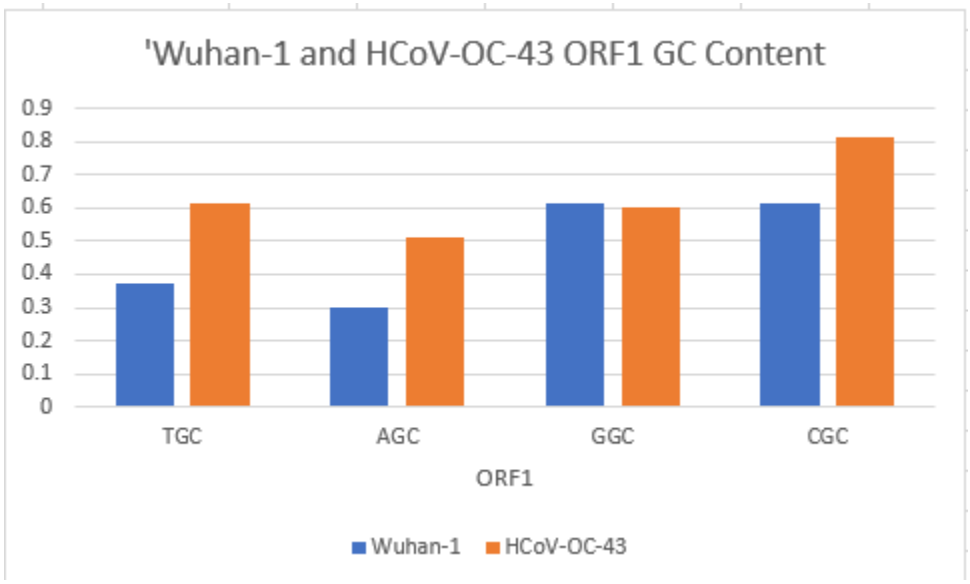


Note: The RSCU gene frequencies of the optimal codons within the ORF1 gene were plotted for comparison which showed Wuhan-1 and RaTG13 are highly correlated whereas HCoV-43 appears less correlated to Wuhan-1.

Next, the GC content was calculated for the codons within the ORF1 gene for SARS-CoV-2 and HCoV-OC43 as seen in Figure 18. The trends observed showed that HCoV-OC-43 has higher GC content in the ORF1 gene compared to the reference strain. This is important as the GC content of synonymous codons has an extreme effect on bias.

Figure 18

Comparison of GC content of Wuhan-1 and HCoV-OC-43 in ORF1



Note: HCoV-oc-43 exhibited codon bias patterns that did not correlate with the reference strain such as higher GC content at below optimal codon frequency positions. The data analyzed supports the trends observed in CoVs in that they have more AT% showing higher codon bias than GC% in genetic codon compositions.

In conclusion, these are just a few examples of analysis methods used in examining codon usage patterns. Understanding codon usage bias can provide insight into the evolution of viruses and their adaptation to other hosts. The data analyzed supports the trends observed in CoVs in that they have more AT% showing higher codon bias than GC% in genetic codon compositions. The frequencies of RSCU genes examined in the SARS-CoV-2 reference, RaTG13, and Banal strains exhibited similar codon bias patterns. HCoV-OC-43 exhibited codon bias patterns that did not correlate with the reference strain such as higher GC content at below optimal codon frequency positions. The overall codon usage pattern of SARS-CoV-2 was initially confirmed to be comparable to the non-human coronaviruses explored in BANAL strains and the RaTG13 strain. Understanding the mutational constraints of these CoVs can provide insight into their adaptability in other hosts.

Discussion

Although multiple researchers worldwide have attempted to answer the question as to if the virus has evolved from its historically known animal hosts and shifted into a human pathogenic virus via naturally occurring mutation or human genetic engineering, the answer to the question remains at large. Nonetheless, the data shown in the various figures reported here demonstrate codon usage among groups of different coronavirus hosts in a single view compared to one another and to the Wuhan-Hu-1 strain. Here, we report codon usage almost exactly the same in strains isolated from the same host species though differences are observed in strains isolated from different host species with the exception of codon utilization in the bat, pangolin, and human strains.

Interestingly, although we find codon usage similarities among strains from the same host species and differences from strains of different host species, the most similar codon usage patterns observed among different hosts include strains isolated from bats, pangolins, and humans. This is particularly evident in Figures 12 and 13 which demonstrate consensus with Mallapaty et al., 2021 showing codon usage among the viral hosts from the older RaTG13 bat strain, to those isolated from bats in Laos, to a Pangolin strain, down to the first reported human strain. We report subtle differences in usage in the three different hosts with a maximum difference of 6% in codon ATA encoding I (isoleucine) in the Pangolin strain (35%) and the RaTG13 strain (29%). These depictions support a natural flow of virial hosts and minimal shifts in codon usage from bats, to pangolins to humans. On the other hand, when observing the differences in codon utilization among the various different hosts in Figure 9, it's quite obvious there's a bias among these different hosts despite some literature suggesting otherwise (Hou et

al., 2020). Taken together, one can easily speculate the most likely transmission events that have occurred in coronavirus hosts which resulted in the virus becoming a human pathogen. Due to this important fact, many of the elements of this project have been researched previously, but that has just allowed us as a group to build upon it and make it better.

In the article *Comparative analysis of human coronaviruses focusing on nucleotide variability and synonymous codon usage patterns* conducted in 2021, researchers analyzed the variability of the sequences of nucleotides and subsequent codon patterns. It was found that in human coronaviruses, the highest GC-content variability was observed in the third codon position, but there have been mutations in all codon locations. Through phylogenetics, this study also showed a great deal of similarity to the Pangolin-CoV which we also observed to be true.

While we chose Python and R to analyze our publicly available data, in another study titled *Characterization of codon usage pattern in SARS-CoV-2* published in late 2020, researchers used CAIcal and COUSIN programs to analyze codon variations within human and non-human coronaviruses. They found that the newer or emergent strains show a lower codon usage bias, meaning that certain codons are not typically more frequently observed than other synonymous codons while gene translation is occurring.

In a 2021 article titled, *Genome-Wide Analysis of Codon Usage Patterns of SARS-CoV-2 Virus Reveals Global Heterogeneity of COVID-19*, it was found that after analyzing the codon usage bias of sixty strains, there was great diversity in the strains based on geographical location. This is something we expected to find and did find in our research between variants such as the South African variant and UK variant. They concluded that while the genome of *SARS-CoV-2 was mainly affected by mutations,* marginal selection pressure when observing codon patterns.

The future implications of this project are huge, if we can accurately map codon variability in SARS-CoV-2, then we can also begin to make prediction models so that we can be better equipped to tailor new vaccines as the ones we have today become obsolete due to mutation. Since the code is already written, the possibilities are endless for what kinds of data we could analyze.

References

Banerjee, T., Gupta, S.K., & Ghosh, T.C. (2005). Towards a resolution on the inherent methodological weakness of the “effective number of codons used by a gene”. *Biochemical and Biophysical Research Communications, 330*(4). 1015-1018. https://doi.org/10.1016/j.bbrc.2005.02.150

CDC. (2022). *About Variants*. COVID-19. https://www.cdc.gov/coronavirus/2019-ncov/variants/about-variants.html

Dilucca, M., Forcelloni, S., Georgakilas, A.G., Giansanti, A., & Pavlopoulou, A. (2020). Codon Usage and Phenotypic Divergences of SARS-CoV-2 Genes. *Viruses, 12*(5). 498. https://doi.org/10.3390/v12050498

Fuglsang, A. (2004). The ‘effective number of codons’ revisited. *Biochemical and Biophysical Research Communications, 317*(3). 957-964. https://doi.org/10.1016/j.bbrc.2004.03.138

Fuglsang, A. (2006). Estimating the "Effective Number of Codons": The Wright Way of Determining Codon Homozygosity Leads to Superior Estimates. *Genetics, 172*(2). 1301-1307. https://doi.org/10.1534/genetics.105.049643

Hou, W. (2020) Characterization of codon usage pattern in SARS-CoV-2.*Virololgy Journal, 17*. 138. https://doi.org/10.1186/s12985-020-01395-x

Kadam, S.B., Sukhramani, G.S., Bishnoi, P., Pable, A.A., & Barvkar, V.T. (2021). SARS-CoV-2, the pandemic coronavirus: Molecular and structural insights. *Journal of Basic Microbiology, 61*(3). 180-202. https://doi.org/10.1002/jobm.202000537

Khattak, S., Rauf, M. A., Zaman, Q., Ali, Y., Fatima, S., Muhammad, P., Li, T., Khan, H. A., Khan, A. A., Ngowi, E. E., Wu, D. D., & Ji, X. Y. (2021). Genome-Wide Analysis of Codon Usage Patterns of SARS-CoV-2 Virus Reveals Global Heterogeneity of COVID-19. *Biomolecules*, *11*(6), 912. https://doi.org/10.3390/biom11060912

Kumar Das, J., & Roy, S. (2021) Comparative analysis of human coronaviruses focusing on nucleotide variability and synonymous codon usage patterns. *Genomics 113*(4). 2177-2188. ISSN 0888-7543. https://doi.org/10.1016/j.ygeno.2021.05.008.

Li, J., Lai, S., Gao, G.F, & Shi, W. (2021). The emergence, genomic diversity and global spread of SARS-CoV-2. *Nature, 600*. 408-418. https://doi.org/10.1038/s41586-021-04188-6

Naqvi, A. A. T., Fatima, K., Mohammad, T., Fatima, U., Singh, I. K., Singh, A., Atif, S. M., Hariprasad, G., Hasan, G. M., & Hassan, Md. I. (2020). Insights into SARS-CoV-2 genome, structure, evolution, pathogenesis and therapies: Structural genomics approach. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease*, *1866*(10), 165878. https://doi.org/10.1016/j.bbadis.2020.165878

NatureEducation. (2010). *The Origins of Viruses*. Scitable. https://www.nature.com/scitable/topicpage/the-origins-of-viruses-14398218/

NatureEducation. (2014). *Genetic Code*. Scitable. https://www.nature.com/scitable/definition/genetic-code-13/

Mallapaty, S. (2021). Closest known relatives of virus behind COVID-19 found in Laos. *Nature*, *597*(7878), 603-603. https://doi.org/10.1038/d41586-021-02596-2

Schoeman, D., Gordon, B., & Fielding, B.C. (2021). Pathogenic Human Coronaviruses. *Reference Module in Biomedical Sciences*, B978-0-12-818731-9.00052-5. https://doi.org/10.1016/B978-0-12-818731-9.00052-5

Sharp, P.M. & Li, W-H. (1987). The codon adaptation index – a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Research, 15*(3). 1281-1295. https://doi.org/10.1093/nar/15.3.1281

Singh, D. & Yi, S.V. (2021). On the origin and evolution of SARS-CoV-2. *Experimental & Molecular Medicine.* 53. 537–547. https://doi.org/10.1038/s12276-021-00604-z7

Sun, X., Yang, Q., Xia, X. (2013). An Improved Implementation of Effective Number of Codons (Nc). *Molecular Biology and Evolution, 30*(1). 191-196. https://doi.org/10.1093/molbev/mss201

Tang, X., Wu, C., Li, X., Song, Y., Yao, X., Wu, X., Duan, Y., Zhang, H., Wang, Y., Qian, Z., Cui, J., & Lu, J. (2020). On the origin and continuing evolution of SARS-CoV-2. *National Science Review*, *7*(6), 1012–1023. https://doi.org/10.1093/nsr/nwaa036

Tort, F.L., Castells, M., & Cristina, J. (2020). A comprehensive analysis of genome composition and codon usage patterns of emerging coronaviruses. *Virus Research*, 283. 197976. https://doi.org/10.1016/j.virusres.2020.197976

Wright, F. (1990). The 'effective number of codons' used in a gene. *Gene, 87*(1). 23-29. https://doi.org/10.1016/0378-1119(90)90491-9

Appendix

Python code

```
# ================================================================================
# Group 2 COVID Analysis Project
# BIOT 670, Spring 2022
# University of Maryland Global Campus
# ================================================================================

# Please refer for calculations:
# Sharp, P. M., & Li, W.-H. (1987). Nucleic Acids Research, 15(3), 1281–1295.
# https://doi.org/10.1093/nar/15.3.1281

# ================================================================================
# Imports
# ================================================================================
from Bio import SeqIO
from Bio.Seq import Seq
from math import exp, log
import pandas as pd
# ================================================================================


# Define amino acids for each codon
gencode = {
'ATA':'I', 'ATC':'I', 'ATT':'I', 'ATG':'M',
'ACA':'T', 'ACC':'T', 'ACG':'T', 'ACT':'T',
'AAC':'N', 'AAT':'N', 'AAA':'K', 'AAG':'K',
'AGC':'S', 'AGT':'S', 'AGA':'R', 'AGG':'R',
'CTA':'L', 'CTC':'L', 'CTG':'L', 'CTT':'L',
'CCA':'P', 'CCC':'P', 'CCG':'P', 'CCT':'P',
'CAC':'H', 'CAT':'H', 'CAA':'Q', 'CAG':'Q',
'CGA':'R', 'CGC':'R', 'CGG':'R', 'CGT':'R',
'GTA':'V', 'GTC':'V', 'GTG':'V', 'GTT':'V',
'GCA':'A', 'GCC':'A', 'GCG':'A', 'GCT':'A',
'GAC':'D', 'GAT':'D', 'GAA':'E', 'GAG':'E',
'GGA':'G', 'GGC':'G', 'GGG':'G', 'GGT':'G',
'TCA':'S', 'TCC':'S', 'TCG':'S', 'TCT':'S',
'TTC':'F', 'TTT':'F', 'TTA':'L', 'TTG':'L',
'TAC':'Y', 'TAT':'Y', 'TAA':'_', 'TAG':'_',
'TGC':'C', 'TGT':'C', 'TGA':'_', 'TGG':'W'}

# Define synonymous_codons
synonymous_codons = {
"C": ["TGT", "TGC"],
"D": ["GAT", "GAC"],
"S": ["TCT", "TCG", "TCA", "TCC", "AGC", "AGT"],
"Q": ["CAA", "CAG"],
"M": ["ATG"],
"N": ["AAC", "AAT"],
```

```
"P": ["CCT", "CCG", "CCA", "CCC"],
"K": ["AAG", "AAA"],
"_": ["TAG", "TGA", "TAA"],
"T": ["ACC", "ACA", "ACG", "ACT"],
"F": ["TTT", "TTC"],
"A": ["GCA", "GCC", "GCG", "GCT"],
"G": ["GGT", "GGG", "GGA", "GGC"],
"I": ["ATC", "ATA", "ATT"],
"L": ["TTA", "TTG", "CTC", "CTT", "CTG", "CTA"],
"H": ["CAT", "CAC"],
"R": ["CGA", "CGC", "CGG", "CGT", "AGG", "AGA"],
"W": ["TGG"],
"V": ["GTA", "GTC", "GTG", "GTT"],
"E": ["GAG", "GAA"],
"Y": ["TAT", "TAC"]}

def gb_parser(file):
# Return biopython genbank record from a genbank formatted file

    gb_record = SeqIO.read(open(file,"r"), "genbank")
    return gb_record


def in_locs(gene_locs, test):
# Determine if a locus (test) falls within a set of gene loci

    found = False
    for i in gene_locs:
        if test in i:
            found = True
    return found


def genes(gb_record):
# From a genbank record, return a list of all CDS names, loci, and sequences

    names = []
    locs = []
    seqs = []
    for feature in gb_record.features:
        # Extract gene names, loci, and sequences from gb record
        if feature.type == "CDS":
            try:
                name = feature.qualifiers.get("gene")[0]
            except TypeError:
                # If gb record does not include gene names, use product name
                name = feature.qualifiers.get("product")[0]
            if name in names:
                # Avoid duplicate entries
                continue
            if (in_locs(locs, feature.location.start) == True and in_locs(locs,
                feature.location.end) == True):
                    # Avoid duplicating sequences (i.e. ORF1a & ORF1ab)
                    continue
```

```
            names.append(name)
            locs.append(feature.location)
            seqs.append(feature.location.extract(gb_record).seq)

    # Assemble list of lists containing all names, loci, and sequences
    genes_all = [names, locs, seqs]
    return genes_all


def cds_codons(cds_list):
# Accept a list of CDS sequences and return as a list of codons

    cds_codons = []
    for cds in cds_list:
        for i in range(0,len(cds),3):
            cds_codons.append(cds[i:i+3])
    return cds_codons


def cds_divide(cds_codons, x):
# Accept a list of codons, return as a list of sequence fragments of x length

    cds_fragments = []
    i=0

    # Split input list of codons into lists of x codons each
    while i < len(cds_codons):
        cds_fragments.append(cds_codons[i:i + x])
        i += x

    # Join codons into sequences
    for f in range(0,(len(cds_fragments))):
        cds_fragments[f] = Seq("").join(cds_fragments[f])

    return cds_fragments


def codon_count(seq):
# Accept a sequence and return codon counts as a dictionary

    codon_dict = {
    "TTT": 0, "TTC": 0, "TTA": 0, "TTG": 0,
    "CTT": 0, "CTC": 0, "CTA": 0, "CTG": 0,
    "ATT": 0, "ATC": 0, "ATA": 0, "ATG": 0,
    "GTT": 0, "GTC": 0, "GTA": 0, "GTG": 0,
    "TAT": 0, "TAC": 0, "TAA": 0, "TAG": 0,
    "CAT": 0, "CAC": 0, "CAA": 0, "CAG": 0,
    "AAT": 0, "AAC": 0, "AAA": 0, "AAG": 0,
    "GAT": 0, "GAC": 0, "GAA": 0, "GAG": 0,
    "TCT": 0, "TCC": 0, "TCA": 0, "TCG": 0,
    "CCT": 0, "CCC": 0, "CCA": 0, "CCG": 0,
    "ACT": 0, "ACC": 0, "ACA": 0, "ACG": 0,
    "GCT": 0, "GCC": 0, "GCA": 0, "GCG": 0,
    "TGT": 0, "TGC": 0, "TGA": 0, "TGG": 0,
```

```
    "CGT": 0, "CGC": 0, "CGA": 0, "CGG": 0,
    "AGT": 0, "AGC": 0, "AGA": 0, "AGG": 0,
    "GGT": 0, "GGC": 0, "GGA": 0, "GGG": 0}

    for i in range(0,len(seq),3):
        codon_dict[seq[i:i+3]] += 1
    return codon_dict


def codon_aafreq(codon_counts):
# Calculate codon frequencies per amino acid from dictionary of codon counts

    codon_aafreq = {
    "TTT": 0, "TTC": 0, "TTA": 0, "TTG": 0,
    "CTT": 0, "CTC": 0, "CTA": 0, "CTG": 0,
    "ATT": 0, "ATC": 0, "ATA": 0, "ATG": 0,
    "GTT": 0, "GTC": 0, "GTA": 0, "GTG": 0,
    "TAT": 0, "TAC": 0, "TAA": 0, "TAG": 0,
    "CAT": 0, "CAC": 0, "CAA": 0, "CAG": 0,
    "AAT": 0, "AAC": 0, "AAA": 0, "AAG": 0,
    "GAT": 0, "GAC": 0, "GAA": 0, "GAG": 0,
    "TCT": 0, "TCC": 0, "TCA": 0, "TCG": 0,
    "CCT": 0, "CCC": 0, "CCA": 0, "CCG": 0,
    "ACT": 0, "ACC": 0, "ACA": 0, "ACG": 0,
    "GCT": 0, "GCC": 0, "GCA": 0, "GCG": 0,
    "TGT": 0, "TGC": 0, "TGA": 0, "TGG": 0,
    "CGT": 0, "CGC": 0, "CGA": 0, "CGG": 0,
    "AGT": 0, "AGC": 0, "AGA": 0, "AGG": 0,
    "GGT": 0, "GGC": 0, "GGA": 0, "GGG": 0}

    for codon in codon_counts:
        # Find total counts for all synonymous codons
        sumni = 0
        for aa in synonymous_codons:
            if codon in synonymous_codons[aa]:
                for i in synonymous_codons[aa]:
                    sumni += codon_counts[i]
        # Calculate frequency for each synonymous codon
        if sumni > 0:
            codon_aafreq[codon] = codon_counts[codon] / sumni
        else:
            # Avoid divide by zero error
            codon_aafreq[codon] = 0

    return codon_aafreq


def get_RSCU(codon_freq):
# Calculate RSCU values from a dictionary of codon frequencies

    rscu = {}
    for val in synonymous_codons.values():
        # Calculate expected frequency if equal usage among synonymous codons
        exp_freq = 1/len(val)
```

```
        # Calculate rscu (observed/expected)
        for i in val:
            rscu[i] = codon_freq[i]/exp_freq

    return rscu


def codon_to_num(codon_dict):
# Accept a dictionary with codons as keys and convert codons to numbers

    # Define numerical values for codons
    codon_num = {
    'ATA':1, 'ATC':2, 'ATT':3, 'ATG':4,
    'ACA':5, 'ACC':6, 'ACG':7, 'ACT':8,
    'AAC':9, 'AAT':10, 'AAA':11, 'AAG':12,
    'AGC':13, 'AGT':14, 'AGA':15, 'AGG':16,
    'CTA':17, 'CTC':18, 'CTG':19, 'CTT':20,
    'CCA':21, 'CCC':22, 'CCG':23, 'CCT':24,
    'CAC':25, 'CAT':26, 'CAA':27, 'CAG':28,
    'CGA':29, 'CGC':30, 'CGG':31, 'CGT':32,
    'GTA':33, 'GTC':34, 'GTG':35, 'GTT':36,
    'GCA':37, 'GCC':38, 'GCG':39, 'GCT':40,
    'GAC':41, 'GAT':42, 'GAA':43, 'GAG':44,
    'GGA':45, 'GGC':46, 'GGG':47, 'GGT':48,
    'TCA':49, 'TCC':50, 'TCG':51, 'TCT':52,
    'TTC':53, 'TTT':54, 'TTA':55, 'TTG':56,
    'TAC':57, 'TAT':58, 'TAA':59, 'TAG':60,
    'TGC':61, 'TGT':62, 'TGA':63, 'TGG':64}

    num_dict = {}
    for key in codon_dict.keys():
        num_dict[codon_num[key]] = codon_dict[key]

    return num_dict


def file_writer(any_dict, name):
# Write dictionary items into a file

    with open(name + ".csv","w") as f:
        for key, val in any_dict.items():
            f.write(str(key) + "," + str(round(val,3)) + "\n")
    print(name + ".csv is written!")


def file_writer2(sample_frags, name):
# Write multiple dictionary items into the same file

    key_list=[]
    freq_list=[]
    for key, val in sample_frags.items():
        key_list.append(key) #alternate 1
        freq_list.append(val)
```

```
    df = pd.DataFrame(freq_list, index=key_list)
    df.to_csv(name + ".csv", sep=",")
    print (name + ".csv is written!")


# ============================================================================

def main():
# From a gb file, calculate RSCU for whole genome and for each gene, output to
# csv. Option to also use fragments of arbitrary size (commented out).

    # Get genome sequence from gb file
    file_name = input("Please enter your GB filename: ")


    # Parse genbank file
    sample_gb_record = gb_parser(file_name)
    sample_genes = genes(sample_gb_record)
    sample_codons = cds_codons(sample_genes[2])


    # Analyze whole genome
    counts_genome = codon_count(Seq("").join(sample_genes[2]))
    freq_genome = codon_aafreq(counts_genome)
    rscu_genome = get_RSCU(freq_genome)


    # Analyze by gene
    freq_genes = {}
    rscu_genes = {}
    for i in range(0,len(sample_genes[0])):
        name = sample_genes[0][i]
        count = codon_count(sample_genes[2][i])
        freq = codon_aafreq(count)
        rscu = get_RSCU(freq)
        freq_genes[name] = freq
        rscu_genes[name] = rscu


##    # Analyze by fragments of any number of codons
##    frag_number = int(input("To divide genome into regions of arbitrary size, "
##                            "enter the desired number of codons per region: "))
##    # Divide genome into fragments and calculate RSCU
##    sample_fragments = cds_divide(sample_codons, frag_number)
##    freq_frags = {}
##    rscu_frags = {}
##    pos = 1
##    for i in range(0,len(sample_fragments)):
##        count = codon_count(sample_fragments[i])
##        freq = codon_aafreq(count)
##        rscu = get_RSCU(freq)
##        freq_frags[str(pos)+"-"+str(pos+frag_number-1)] = freq
##        rscu_frags[str(pos)+"-"+str(pos+frag_number-1)] = rscu
```

```
##        pos += frag_number


    # Option to convert codons to numbers
    convert = input("Convert codons to numeric values? Enter Y/N: ")
    if convert.lower() == "y":
        rscu_genome = codon_to_num(rscu_genome)
        for key in rscu_genes.keys():
            rscu_genes[key] = codon_to_num(rscu_genes[key])
##        for key in rscu_frags.keys():
##            rscu_frags[key] = codon_to_num(rscu_frags[key])


    # Write genome rscu table to csv file
    file_writer(rscu_genome, file_name + "_rscu_genome")

    # Write gene rscu table to csv file
    file_writer2(rscu_genes, file_name + "_rscu_genes")

##    # Write fragment rscu table to csv file
##    file_writer2(rscu_frags, file_name + "_rscu_frags")


# ============================================================================

if __name__ == "__main__":
    main()
```

R code

R code for visualization of codon frequencies of different SARS-CoV-2 viral strains:

```
install.packages("ggplot2")

library(ggplot2)

codon_usage <- read.csv("Wuhan_codon_list.csv")

ggplot(codon_usage, aes(x=Codon, y=Frequency, fill = AA)) +

facet_wrap(~ AA, scale="free_x") + theme_bw()+

geom_bar(stat = "identity")+

ylab("Codon Usage Frequency")+xlab("Codon")+

labs(title = "Wuhan-Hu-1 Codon Frequency Per Amino Acid")+

theme(axis.text.x = element_text(size = 7))+

theme(legend.position="none")
```

R code for visualization of RSCU values of specific genes and gene fragments for the different SARS-CoV-2 viral strains :

```
install.packages("ggplot2")

library(ggplot2)

install.packages("tidyr")

library(tidyr)

codon_usage <- read.csv("test1.csv")

ggplot(codon_usage, aes(x=Codon, y=RSCU_Values, fill = Measure))+

geom_bar(stat='identity')+

facet_wrap(~Measure, ncol=1, strip.position = "left")+

theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+

theme(legend.position="none")+

labs(title = "input_graph_title_here")
```