PharmaSUG 2022 - Paper AD-167

Using the SAS System® with the National Center for Biotechnology Information resources and Immune Epitope Database to explore the realm of potential SARS-CoV-2 variants.

Kevin R. Viel, Ph.D., Navitas Data Sciences; Histonis, Incorporated

ABSTRACT

SARS-Cov-2 emerged in late 2019 and quickly became a pandemic. Infection may result in Covid-19, which may cause acute respiratory distress syndrome (ARDS), a risk factors for death, and post-infection sequelae (Covid-19 long haulers). The public databases of the National Center for Biotechnology Information (NCBI) of the United States National Institute of Health (NIH) include RNA sequences of SARS-CoV-2 and their classification by the Pango Nomenclature System.

The number of viruses produced by an infected person and the poor fidelity of replication of viruses make variants of SARS-CoV-2 a practical certainty. The “epitope escape” model may predict which variants emerge and, specifically, does not require inter-human transmission. Even asymptomatic infections will generate variants. Generally, an MHC-II groove must bind a peptide with sufficient affinity to effectively present it to the adaptive immune system to generate antibodies. Tools such as the Immune Epitope Database (IEDB) estimate which peptides might bind to the MHC-II molecules.

The goal of this paper is to introduce the NCBI repository, to demonstrate the use the SAS System® to enumerate the realm of variants expected in a given SARS-CoV-2 sequence, i.e. the lineages, that may arise, given a single, random nucleotide substitution per codon and then estimate the binding affinity of the resulting peptides to the MHC-II genotypes available in IEDB. This paper demonstrates snippets of SAS code that are required given the sheer size of genomic data.

INTRODUCTION

Microbes are a facet of life, whether their hosts are single cell organisms or complex, multicellular animals with hundreds of types of specialized cells comprising tissues that constitute organs. A non-exhaustive list of the types of microbes includes bacteria, archaea, fungi (mold and yeast), protozoans (Plasmodium falciparum, a cause of malaria), and viruses, all of which may influence human health and evolution. For instance, bacteria can be pathogenic and further bacteria resistant to antibiotics are a considerable public health challenge. Even in the United States, sepsis is a leading cause of death. Bacterial infections may be classified as nosocomial (hospital-acquired) or community-based and some genetic elements that may confer resistance may help to distinguish between the source of the infections. Microbes may also be commensal, such as many bacteria and archaea that comprises some of the flora of the human gut.

Given that “bacteria alone [are] estimated to outnumber human cells within an individual by an order of magnitude,”[1] the Human Microbiome Project was formed to provide a framework for data collection, data sharing, and research[1,2]. The number of viruses far outpaces the number of bacteria and the metagenomic investigations[3] of them is fascinating due to both the technologcial and computation resources required and the diverse environments under investigation, such as coral reefs or lungs of patients with cystic fibrosis; the investigations of what factors contribute to success in niche constitutes an incredible new era of discovery.

Some coronaviruses infect humans, for example, HCoV-229E, HCoV- OC43, HCoV- NL63, and HCoV- HKU1 cause the “common cold”[4] and may re-infect individuals “seasonally”. In fact, Galanti and Shaman[5] reported that in their investigation of 191 participants, “12 individual tested positive multiple times for the same coronavirus”[5] within an active surveillance period of a year, raising the question of how the viruses might be evading the innate and adaptive immune systems. Recently, novel coronaviruses emerged as deadly, but short-lived epidemics: SARS-CoV-1 (**S**erious **A**cute **R**espiratory **S**ydrome and

COrona**V**irus) and MERS (**M**iddle **E**astern **R**espiratory **S**ydrome). In December 2019, SARS-CoV-2 emerged and was quickly designated as a pandemic.

An individual infected with SARS-CoV-2, "carries 10^9 to 10^{11} virions during peak infection"[6]. The sheer number of viruses replicated implies that variants will arise by random error in replication, that is a substitution, with an unknown and perhaps variant probability, of a nucleotide, i.e., a single nucleotide variant, among other types of variants (Table 1 of Viel[7] and Table 3 of Viel[8]). Even "asymptomatic" people in whom viruses replicate may be reservoirs of (new) variants that arise from de novo mutations during the replication process.

In describing active hepatitis C infections, Salloum et elia[9] reasoned that "[w]ith these error rates—and a high turnover rate of an estimated 10^{12} virions per day (Neumann[10] as quoted by Salloum et alia[9])—theoretically, all possible mutations in every single position of the genome will be generated in one infected host every day." The process of replication of the RNA sequences may incur varying "costs" to the cell, such as codon usage, folding, trafficking, et cetera covered under the general term of "fitness" costs. In addition, a potential selective pressure relating to the resulting variant proteins for which the RNA codes is "the binding energy between Receptor-Binding Domain (RBD) and Angiotensin-Converting Enzyme 2 (ACE2)"[11], but will not be discussed in this paper. As translation in the cell produces the proteins required to constitute a viable virus, the cell may cleave those proteins into peptides that MHC-I and MHC-II complexes may bind and present to immune cells. If a receptor on those cells, such as a T cell binds the MHC-peptide complex, then the adaptive immune system might also add a selective pressure, a concept described as epitope escape.

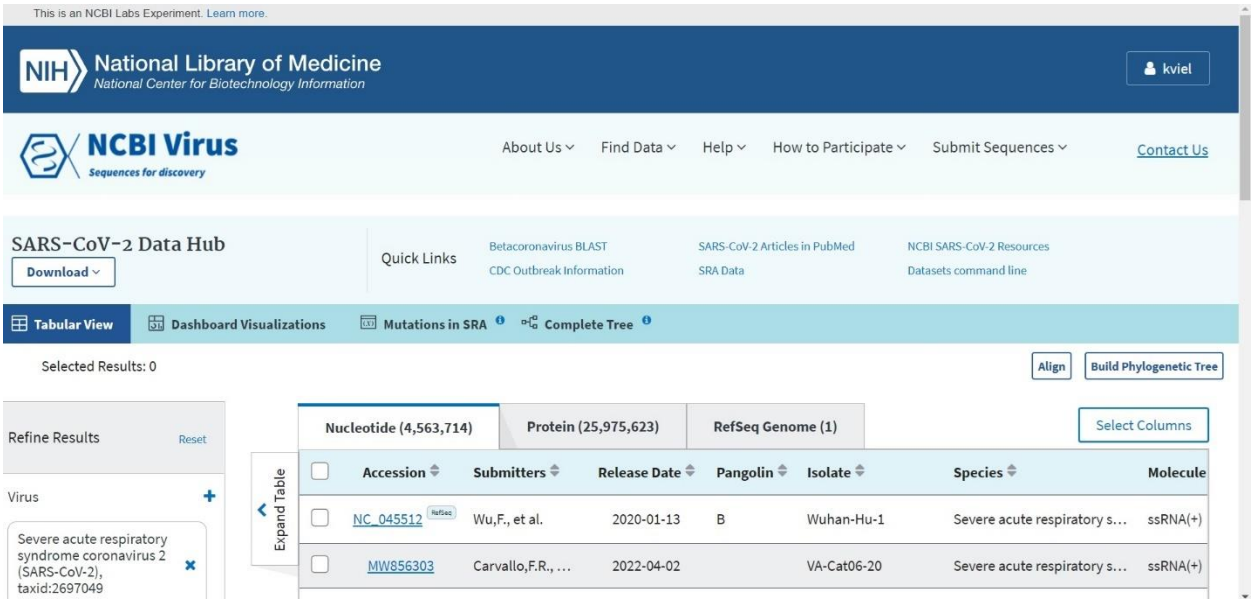
Importantly, with the number of infected individuals and the distribution of select factors, like the genotypes of MHC, that the same variants of concern may arise de novo in separate populations without transmission between them is entirely possible. Without active and adequate surveillance, we may not know who was exposed to what, except that we know that infected individuals were exposed, but we may not know the genotypes of the viruses that successfully entered their bodies, reach cells with appropriate receptors (ACE-2), and replicated. In fact, given this general discussion, we might be surprised if samples taken from infected patients did not result in hundreds of identified variants.

The National Center for Biotechnology Information (NCBI) of the National Institutes of Health (NIH) of the United States provides numerous free resources for genomic research, including the SARS-CoV-2 Data Hub[12]. Importantly, this is one resource for sequences of SARS-CoV-2 submitted by the community. The National Institute of Allergy and Infectious Diseases (NIAID) of the US NIH funds the Immune Epitope Database[13], which is also available as a free resource.

The **goal** of this paper is to introduce the NCBI repository, to demonstrate the use the SAS System® to enumerate the realm of variants expected in a given SARS-CoV-2 sequence, i.e. the lineages, that may arise, given a single, random nucleotide substitution per codon and then estimate the binding affinity of the resulting peptides to the MHC-II genotypes available in IEDB. This paper demonstrates snippets of SAS code that are required given the sheer size of genomic data.

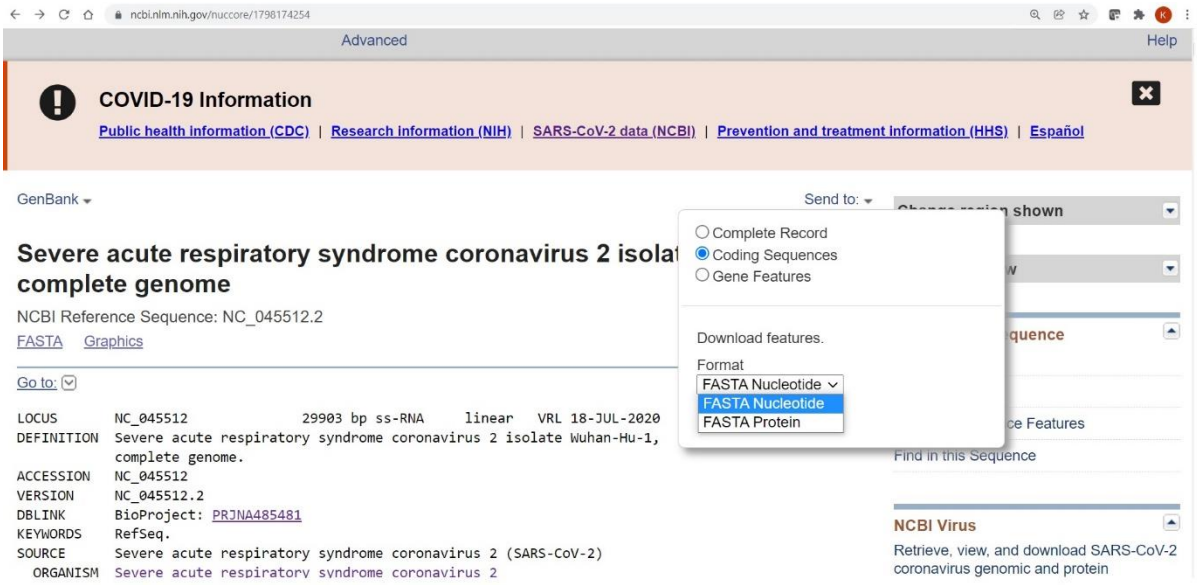
METHODS

The sequences of SARS-CoV-2 were downloaded from the NCBI Virus SARS-CoV-2 Data Hubs (**Display 1**). As of April 03, 2022, the data hub contained 4,563,714 records for nucleotide sequences, about one million more than when the author last downloaded the data, which required multiple hours and was left as an overnight process.



Display 1 The SARS-CoV-2 Data Hub as of April 03, 2022.

The reference sequence for this paper is NCBI Reference Sequence NC045512.2. As a cross-check, the author downloaded the first record using a more familiar NCBI data base to him, the NUCCORE (**Display 2**). The sequences in various formats are obtained by clicking “Send to:” and selecting the desired file, such as FASTA Nucleotide or FASTA Protein. Both were obtained and read into SAS using the code in **Appendix 1**.



Display 2 The NCBI Reference Sequence NC_045512.2 in the NCBI NUCCORE database.

The headers and first lines to the fasta sequences are (note the items in bold are the column headers of Table 1):

FASTA Nucleotide

```
>lcl|NC_045512.2_cds_YP_009724389.1_1 [gene=ORF1ab] [locus_tag=GU280_gp01]
[db_xref=GeneID:43740578] [protein=ORF1ab polyprotein] [exception=ribosomal
slippage] [protein_id=YP_009724389.1]
[location=join(266..13468,13468..21555)] [gbkey=CDS]
ATGGAGAGCCTTGTCCCTGGTTTCAACGAGAAAACACACGTCCAACTCAGTTTGCCTGTTTTACAGGTTC
```

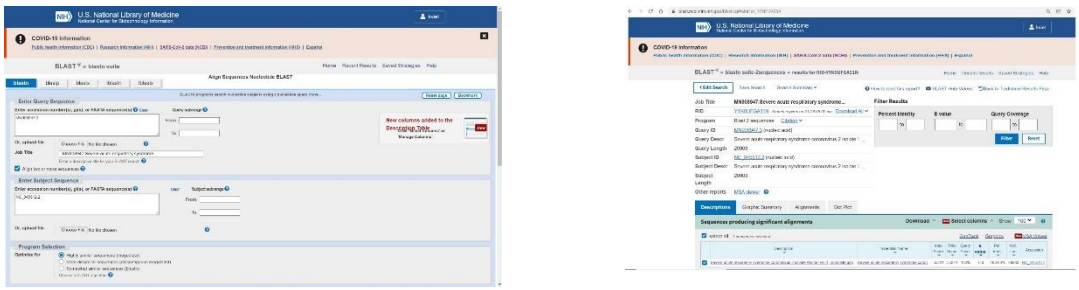
FASTA Protein

```
>lcl|NC_045512.2_prot_YP_009724389.1_1 [gene=ORF1ab] [locus_tag=GU280_gp01]
[db_xref=GeneID:43740578] [protein=ORF1ab polyprotein] [exception=ribosomal
slippage] [protein_id=YP_009724389.1]
[location=join(266..13468,13468..21555)] [gbkey=CDS]
MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHLKDGTCGLVEVEKGVLPQLEQPYVF
```

Note that NCBI updates records, so the exact details may not match (one reason we approach this programmatically, the code can simply be re-run in isolation or as part of an application). To verify that the substitutions of single nucleotides per codon will correspond correct codons, the original nucleotide sequences will be translated to their corresponding protein sequences using the SAS formats in **Appendix 2** with the code in **Appendix 3.** C2AA is Codon to Amino Acid and C2AAS is Codon to Amino Acid Symbol. To manually confirm, the first three codons are ATG, GAG, and AGC. Finding the values on line 143, 119, and 158, respectively, of Appendix 1, we see that the corresponding amino acid symbols are M, E, and S, which match to the first three amino acids in the fasta file.

The file downloaded from the Data Hub contained multiple fasta files, one for each record. This file was read with slight variations. First, in the output data sets, each nucleotide results in one observation. The header was not repeated for each observation, but instead was indexed. The “id” file contains the header and its index. The “seq” file also contains the nucleotide position within the individual fasta files, i.e. each starts with one. This process is a normalization of the data, resulting in an additional data set and the requirement to join the two on ID, but decreases the requirement and allows sequences longer than the maximum length of a SAS character variable to be included (SARS-CoV-2 might be a large virus in terms of the length of its RNA, but it is still quite small; in contrast the human F8 gene is 186 KB.) The code in **Appendix 4** is one way to accomplish this task.

To verify that the sequences from the two sources match, the data corresponding to ID = 1 was used to create a fasta file, which was compared to the original single fasta file using BLAST®[14,15]. **Display 3** demonstrates the alignment of two sequences via the web interface of BLASTn®.



Display 3 The BLASTn web interface and the initial results window.

The manual process of interacting with a website is valuable but cannot be easily automated and requires more time. NCBI has freely available standalone version that is well suited to the needs of this paper. An updated version of the original macro[7] is available upon request. In this case, the code to run the macro is:

```
%m_u_blast
   ( query_id_ds          = SARS_CoV_2_NC_045512_2_id
   , query_seq_ds         = SARS_CoV_2_NC_045512_2_seq
   , subject_id_ds        = SARS_CoV_2_id
   , subject_seq_ds       = SARS_CoV_2_seq
   ) ;
```

GENERATING VARIANTS

A codon consists of three nucleotides in sequence. In each position, one of four nucleotides (A,C,G, and T) are possible for a total of 4^3 = 64 combinations (see Appendix 3). To generate the variants that might arise from one nucleotide substitution, potentially the error in replication that is most likely, at each position of a codon, each of the three nucleotides other than the one in the reference codon was substituted. For instance, for the second codon GAG discussed above, the three other substitutions of the first nucleotide of the codon results in AAG, CAG, and TAG corresponding to K (Lysine), Q (Glutamine), and a premature termination (stop codon) , respectively. For every amino acid, but the first, the start of translation signal, M, the distinct amino acids resulting from a single, non-synonymous nucleotide substitution were retained in a data set with the codon number (amino acid position in the sequence). The code in **Appendix 6** accomplish this task. For the purposes of this paper, on the first sequence (ID = 1, which, as stated above, corresponds to NC045512.2) was used to generate variants.

Visually Inspecting Peptides with Variants

The intended design of the peptides is to match the reference peptide except at one amino acid. The nonsynonymous amino acid (NAA) will occupy each position, 1 through n, if possible. The number of NAAs per position of the full-length protein was summarized. Peptides of lengths 14-16 were generated because those are the length expected to bind the MHC-II complex (approximately nine are expected to bind the groove, which is open in contrast to the groove of MHC-I, and the flanking region is expected to stabilize the binding). Once the peptides were generated with the code in Appendix 6, the following code was used (not that option updating the WORD_SIZE to 5, since exact matches with no gaps were desired *for this task*):

```
%m_u_blast
   ( blast                = blastp
   , query_id_ds          = query_id
   , query_seq_ds         = query_seq
   , subject_id_ds        = subject_id
   , subject_seq_ds       = subject_seq
   , option               = %str(,) " -word_size 5"
   , blast_out_ds         = blastp
   , fdelete              = N
   ) ;
```

BLAST®, to the knowledge of the author, creates graphics for alignments of two and only two sequences. The code in **Appendix 7** takes the results of multiple alignments to the same reference sequence (the anchor sequence) and displays them. A subset of the possible peptides was visually examined to confirm that they matched their design. The author refers the interested reader a related paper[16] that provides the description of the peptides in more detail, such as Figure 2[16]. An example of a partial file with multiple fasta records used is:

```
>ORF10_protein_0001
MGYINVFAFPFTIYSLLLCRMNSRNYIAQVDVVNFNLT
>ORF10_protein_L14_0001
MAYINVFAFPFTIYSILLCRMNSRNYIAQADVVNFNLT
>ORF10_protein_L14_0002
MCYINVFAFPFTIYSPLLCRMNSRNYIAQEDVVNFNLT
```

ESTIMATED BINDING AFFINITY

The immune epitope database (IEDB)[17-18] offers a web interface and standalone tools to generate estimated binding affinities between peptides generated from a sequence and various alleles of MHC-II. The previous paper[16] used the web interface of NetMHCIIpan[19] to manually obtain the estimated binding affinities for variant peptides of coagulation co-enzyme Factor VIII. The scope of this (eventual) project is orders of magnitude greater, not only because we may be examining variants of variants (of variants…), but also because the extension is multiple nonsynonymous substitution per peptide, requiring the elimination of manual approach and the need to work over the internet (also note that for the web interface, NetMHCIIPan imposes limitations). The standalone of versions of IEDB tools required the installation of a virtual machine and Linux, after which the author corresponded with the support desk of the National Library of Medicine (NLM) of the NIH by email. An example of the command line to obtain estimates is:

```
python mhc_II_binding.py consensus3 HLA-DPA1*01/DPB1*04:01,HLA-
DRB1*01:01,HLA-DRB1*04:04\
,HLA-DRB1*08:04,HLA-DRB1*11:21,HLA-DRB1*13:27\
 test.fasta > /media/sf_iedb/test.txt
```

The author highly recommends a (bash) script that includes a loop, but after obtaining the list of available alleles, the reader can use SAS to write the command line for the various fasta files. The output of the command line immediate above can be read into SAS using the code in **Appendix 8**. Note that the M_PGM_IEDB_CONSENSUS3_READ macro calls two macros, M_U_DIRECTORY_READ and M_U_MAXLEN, that are not defined in this paper. They and other macros that they call are available by (email) request, but the first (recursively) reads a directory (path) and list the child files and directories and the second "trims" the lengths of variables to the longest value. That the directory read macro is not required on a Linux system may be obvious to some.

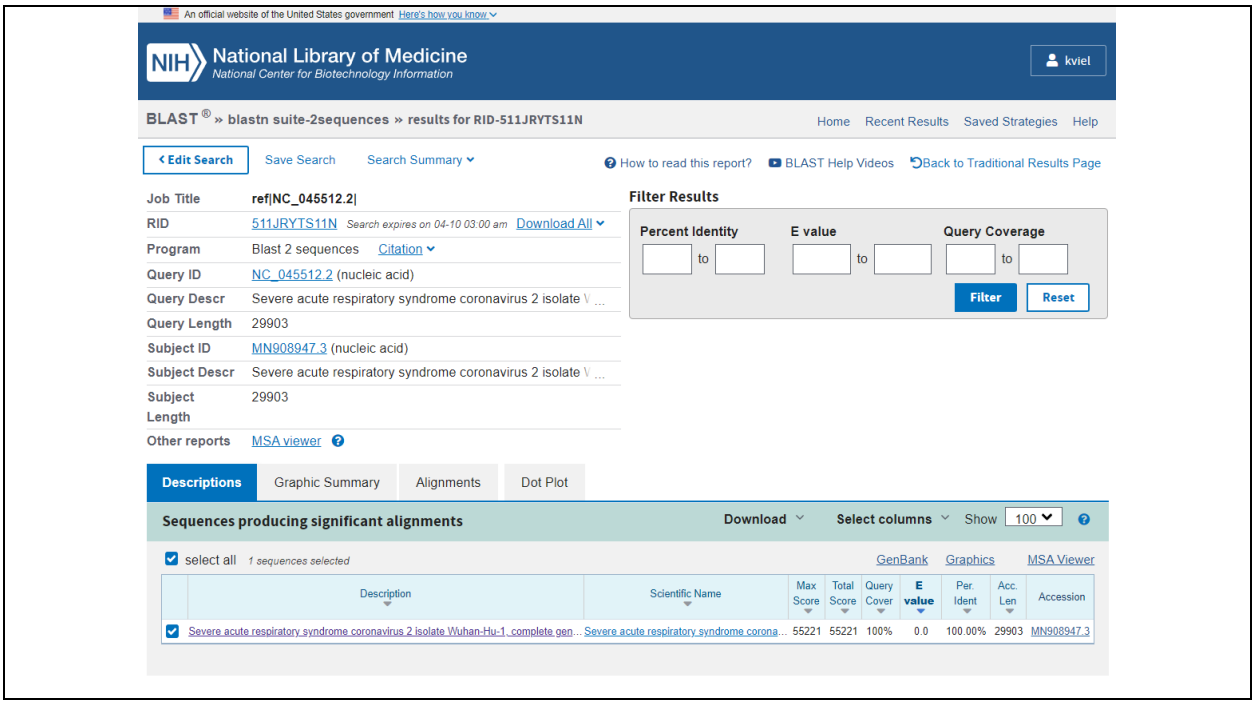
RESULTS

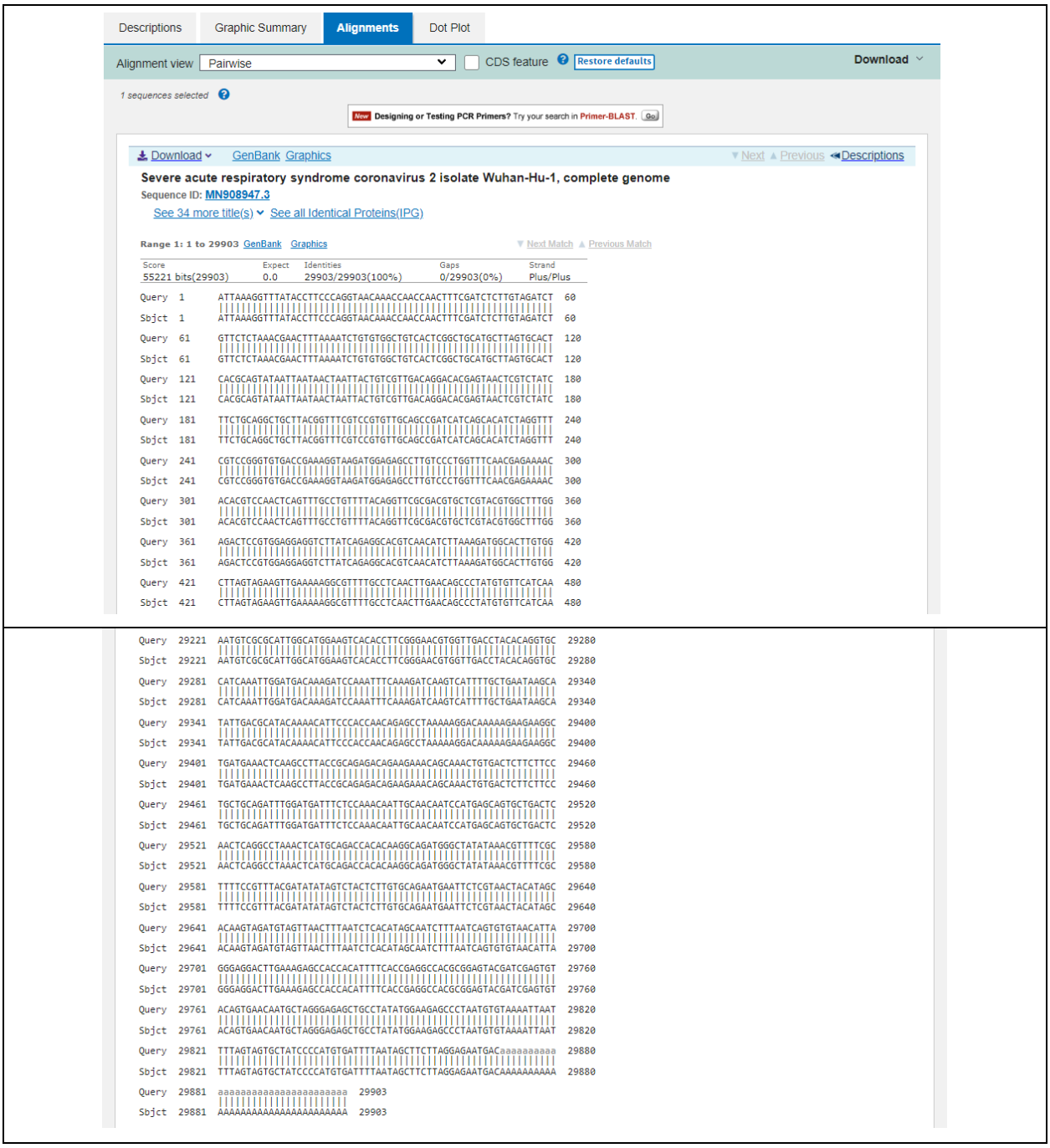
Table 1 summarizes the counts of nucleotides and amino acids in the nucleotide and protein fasta sequences in NC045512.2 (Display 2). The astute reader may observe that the number of nucleotides (bp, base pairs) in Display 1 is 29,903, but the total number of nucleotides in Table 1 far exceeds that value. ORF1a polyprotein overlaps ORF1ab polyprotein among other overlapping segments. Further, the number of nucleotides should be three times the number of amino acids, but a quick look at table one suggests this is not the case. The last codon codes for translation termination, so the number of amino acids is one less (3822 / 3 = 1273 + 1).

Table 1 Summary of the Coding Sequences file for NCBI Reference Sequence NC045512.2

PROTEIN	PROTEIN_ID	LOCATION	NUCLEOTIDES	AMINO ACIDS
surface glycoprotein	YP_009724390.1	21563..25384	3822	1273
ORF3a protein	YP_009724391.1	25393..26220	828	275
envelope protein	YP_009724392.1	26245..26472	228	75
membrane glycoprotein	YP_009724393.1	26523..27191	669	222
ORF1a polyprotein	YP_009725295.1	266..13483	13218	4405
ORF6 protein	YP_009724394.1	27202..27387	186	61
ORF7a protein	YP_009724395.1	27394..27759	366	121
ORF7b	YP_009725318.1	27756..27887	132	43
ORF8 protein	YP_009724396.1	27894..28259	366	121
nucleocapsid phosphoprotein	YP_009724397.2	28274..29533	1260	419
ORF10 protein	YP_009725255.1	29558..29674	117	38
ORF1ab polyprotein	YP_009724389.1	join(266..13468,13468..21555)	21291	7096

Display 4 shows the results of the BLASTn alignment of NC045512.2 and MN908947.3 as an example of the use of the web interface. Although not shown, the first fasta record abstracted from the SARS-CoV-2 Data Hub also exactly aligns with NC045512.2 and confirms the results of using these sequences in the M_U_BLAST macro.





Display 4 A partial display of the results of BLASTn between NC045512.2 and MN908947.3.

Table 2 Summarizes the number of amino acids in each of the SARS-CoV-2 proteins of the NC045512.2 reference sequence that have at least one nonsynonymous substitution. As can be inferred from Table 1, each amino acid in each protein has a nonsynonymous substitution. **Table 3** displays the minimum and maximum number of variant amino acids at one position by each SARS-CoV-2 proteins of the NC045512.2

Table 2 Summarization of the number of amino acids with at least one variant by SARS-CoV-2 protein based on NC045512.2.

LOCATION	PROTEIN	AMINO ACIDS	VARIANTS
21563..25384	surface glycoprotein	1273	1273
25393..26220	ORF3a protein	275	275
26245..26472	envelope protein	75	75
26523..27191	membrane glycoprotein	222	222
266..13483	ORF1a polyprotein	4405	4405
27202..27387	ORF6 protein	61	61
27394..27759	ORF7a protein	121	121
27756..27887	ORF7b	43	43
27894..28259	ORF8 protein	121	121
28274..29533	nucleocapsid phosphoprotein	419	419
29558..29674	ORF10 protein	38	38
join(266..13468,13468..21555)	ORF1ab polyprotein	7096	7096

Table 3 The minimum (MIN) and maximum (MAX) of the number of amino acids occurring at a single amino acid by SARS-CoV-2 protein based on NC045512.2.

LOCATION	PROTEIN	MIN	MAX
21563..25384	surface glycoprotein	1	7
25393..26220	ORF3a protein	1	7
26245..26472	envelope protein	1	7
26523..27191	membrane glycoprotein	1	7
266..13483	ORF1a polyprotein	1	7
27202..27387	ORF6 protein	1	7
27394..27759	ORF7a protein	1	7
27756..27887	ORF7b	1	7
27894..28259	ORF8 protein	1	7
28274..29533	nucleocapsid phosphoprotein	1	7
29558..29674	ORF10 protein	1	7
join(266..13468,13468..21555)	ORF1ab polyprotein	1	7

Table 4 Displays the number of amino acids variants per position by each protein in by SARS-CoV-2 protein based on NC045512.2. Note multiple, distinct nucleotide substitutions can result in the same variant amino acid. **Figure 1** shows the align of the amino acids of ORF10 protein with variant substitutions at every 14^{th}, 15^{th}, and 16^{th} position starting at the second amino acid. The protein sequences were designed so that in peptides of lengths 14-16, only one amino acid was variant and that every variant appeared in peptide of the required length (see Table 3). A variant of the appropriate amino acid was substituted at each off the positions of the peptide.

An example of a subset of results from IEDB for allele HLA-DPA1*01:03/DPB1*02:01 is:

seq_num	start	end	peptide	consensus_ percentile_ rank	adj_consensus_ percentile_ rank	comblib_ core
85	2328	2341	FLAYILFTRFFYVL	0.01	0.01	ILFTRFFYV
85	2329	2342	LAYILFTRFFYVLG	0.01	0.01	ILFTRFFYV
85	2330	2343	AYILFTRFFYVLGL	0.01	0.01	ILFTRFFYV
90	2328	2341	FLAYILFTRFFYVI	0.01	0.01	ILFTRFFYV

seq_num	start	end	peptide	comblib_ score	comblib_ rank	adjusted_ comblib_ rank
85	2328	2341	FLAYILFTRFFYVL	0.01	0.01	0.01
85	2329	2342	LAYILFTRFFYVLG	0.01	0.01	0.01
85	2330	2343	AYILFTRFFYVLGL	0.01	0.01	0.01
90	2328	2341	FLAYILFTRFFYVI	0.01	0.01	0.01

Note that only peptides that meet the binding criteria are output by IEDB. The original (input) file with the fasta headers or a SAS data set with that data are required to obtain the compliment peptides for this investigation. With such data, one can determine if an amino acid variant might be a risk factor for epitope escape, that is, the referent peptide binds to patients MHC-II grooves and, thus, cannot be presented to the T cell receptors, but the peptide with the variant binds with high affinity.

Table 4 The distribution of the number of amino acid variants per position by SARS-CoV-2 protein based on NC045512.2.

LOCATION	PROTEIN	Number of Amino Acid Variant per position			
		4	5	6	7
21563..25384	surface glycoprotein	284	765	4950	1575
25393..26220	ORF3a protein	60	220	1044	301
26245..26472	envelope protein	12	70	306	56
26523..27191	membrane glycoprotein	56	205	774	273
266..13483	ORF1a polyprotein	1000	2955	17394	4662
27202..27387	ORF6 protein	16	30	222	105
27394..27759	ORF7a protein	24	75	534	84
27756..27887	ORF7b	20	30	138	70
27894..28259	ORF8 protein	44	70	444	161
28274..29533	nucleocapsid phosphoprotein	116	315	1590	441
29558..29674	ORF10 protein	.	25	162	49
join(266..13468,13468..21555)	ORF1ab polyprotein	1660	4685	27606	8008

```
Ref   M G Y I N V F A F P F T I Y S L L L C R M N S R N Y I A Q V D V V N F N L T
        |                         |                         |
14    M A Y I N V F A F P F T I Y S I L L C R M N S R N Y I A Q A D V V N F N L T
        |                         |                         |
15    M A Y I N V F A F P F T I Y S L F L C R M N S R N Y I A Q V D A V N F N L T
        |                         |                         |
16    M A Y I N V F A F P F T I Y S L L F C R M N S R N Y I A Q V D V V D F N L T
        - - - - + - - - - 1 - - - - + - - - - 2 - - - - + - - - - 3 - - - - + - -
```

Figure 1 Alignment of the sequences of ORF10 protein with variants every 14, 15, and 16 amino acids starting at the second amino acid with the reference (NC045512.2) ORF10 protein of the SARS-CoV-2 ORF10 protein. Note that this "staggered" placement of results in peptides of the given lengths with one variant amino acid per peptide, in this example, in the first position of the peptides. Also note that the bars indicate a mismatch instead of the BLAST convention of indicating a match. The standard SAS® spacing bar appears in the last line but has spaces padding the position for readability.

CONCLUSION

SARS-CoV-2 emerged as pandemic in early 2020 and the US NIH, among other institutions, responded by providing resources, such as the SARS-CoV-2 Data Hub. This paper demonstrated how to download and use the sequences, a seminal tool to epidemiologic investigations and the development of vaccines and therapeutics. This paper introduced the use of a IEBD tool to estimate the binding affinity of sequences, including known and hypothetical variants. Potentially, emerging variants might be predicted based on the concepts of epitope escape, codon usage bias, and binding energy estimation, among other fields of investigation.

The approach in this paper may or may not have approximated the true diversity of variations of viruses in an infection of a single patient. Whether a multitude of viruses of varying sequence infect a patient seems to be unknown. Whether a virus of a certain sequence that successfully infects a cell of a patient produces viruses with variants in regard to the original sequence seems to be unknown. Whether multiple virus infect the same cell and at what times seems unknown. Given the sheer number of viruses, it seems that all of these three scenarios seems likely. The limitations in technology, i.e. single molecule sequencing with sufficiently large read-lengths, and the lack of sufficient active surveillance would seem to impact the reporting of sequences of viruses. Given the number of viruses estimated to be produced in a single infection and the hundreds of millions of people that have been infected, that SARS-CoV-2 only has under five million records should alert us to a potential shortcoming of surveillance or reporting, or both.

Note SARS-CoV-2 "testing" is *not* sequencing, rather, it is usually RT-PCR (Reverse Transcription Polymerase Chain Reaction). The sample RNA is reverse transcribed ("back") to DNA and then amplified by PCR. PCR requires known and mostly invariant sequences to define the pairs of reverse and forward primers. The qualifier "invariant" pertains to possibility that potential variants in the RNA sequences that correspond to the "ends" of the primers might not disrupt binding, but a gel might reveal "noise". Variants in the sequences corresponding to the primers used in SARS-CoV-2 PCR test might affect binding of the respective primers and result in a negative test. Examining the distribution of variants in sequence classified as regions where primer do and do not bind might reveal a "detection" bias; almost certainly, if a sample is not sent for sequencing if the patient is not positive by PCR test, then we may expect a detection (of sequence variants) issue. This raises the interesting issue in vaccine research and clinical development of not just if the patients were exposed (currently, that is only certainly known for patients who test positive, in most conceivable situations), but also how their MHC-II

genotypes, among others, are factors for Covid-19 and production of viable viruses, including those with variations.

Another possibility to "track" the sequence of variants of variants that might lead to different Pango nomenclature[20] is an interesting investigation. Removing the condition of one nucleotide substitution per codon or one nonsynonymous amino acid per peptide is a computational, but perhaps worthwhile, burden. Determining whether a reference sequence from viruses that infect one species might result from a "path" of "variants of variants" of another species might be interesting, but the SAS programmer who has the computation resources, time, and interest is warned that the same issues of detection and surveillance might be an issue.

This paper variously excluded potential variants in the start and termination codons (first and last codons, the first amino acid always being M). Since variants that do not result in translation of (full-length) proteins may still produce proteins that can be cleaved into peptides that are displayed in MHC-II, thus, impacting immune responses[21], this was probably an oversight. In the least, it may affect the fitness of the original infecting (replicating) virus, since non-viable replicated RNA would be expected to impact lineage of that virus.

This paper raises the issue that the same variants may arise in separate populations without transmission. Certainly, whether travel restrictions might limit the emergence of variants of concerns is *not addressed*, but it might advise non-technical reporting, i.e. the media, to update its language.

If this paper suggests public health or vaccine development platforms, then it reinforces the need for whole-genome sequencing, not only of the viruses obtained from a sample of patient (or every patient), but of the patients themselves. The concept of epitope escape not only relies on MHC-II "type", but also which peptides are likely to be cleaved. Regarding the MHC-II typing, certainly whole-genome sequencing combined with the resulting large population size will enable the next era of research and public health practice.

REFERENCES

[1] Human Microbiome Project Consortium. A framework for human microbiome research. Nature. 2012 Jun 13;486(7402):215-21. doi: 10.1038/nature11209. PMID: 22699610; PMCID: PMC3377744.

[2] Integrative HMP (iHMP) Research Network Consortium. The Integrative Human Microbiome Project. Nature. 2019 May;569(7758):641-648. doi: 10.1038/s41586-019-1238-8. Epub 2019 May 29. PMID: 31142853; PMCID: PMC6784865.

[3] Edwards RA, Rohwer F. Viral metagenomics. Nat Rev Microbiol. 2005 Jun;3(6):504-10. doi: 10.1038/nrmicro1163. PMID: 15886693.

[4] V'kovski P, Kratzel A, Steiner S, Stalder H, Thiel V. Coronavirus biology and replication: implications for SARS-CoV-2. Nat Rev Microbiol. 2021 Mar;19(3):155-170. doi: 10.1038/s41579-020-00468-6. Epub 2020 Oct 28. PMID: 33116300; PMCID: PMC7592455.

[5] Galanti M, Shaman J. Direct Observation of Repeated Infections With Endemic Coronaviruses. J Infect Dis. 2021 Feb 13;223(3):409-415. doi: 10.1093/infdis/jiaa392. PMID: 32692346; PMCID: PMC7454749.

[6] Sender R, Bar-On YM, Gleizer S, Bernshtein B, Flamholz A, Phillips R, Milo R. The total number and mass of SARS-CoV-2 virions. Proc Natl Acad Sci U S A. 2021 Jun 22;118(25):e2024815118. doi: 10.1073/pnas.2024815118. PMID: 34083352; PMCID: PMC8237675.

[7] Viel, K. 2012. "Using the SAS System as a bioinformatics tool: a macro that calls the standalone Basic Local Alignment Search Tool (BLAST) setup." Proceedings of the PharmaSUG 2012 Conference, San Francisco, CA: PharmaSUG. https://www.pharmasug.org/proceedings/2012/HO/PharmaSUG-2012-HO07.pdf

[8] Viel, K. 2011. "Creating reference amplicons and genotyping using the SAS System." Proceedings of the PharmaSUG 2011 Conference, Nashville, TN PharmaSUG. https://www.pharmasug.org/proceedings/2011/AD/PharmaSUG-2011-AD16.pdf

[9] Salloum S, Oniangue-Ndza C, Neumann-Haefelin C, Hudson L, Giugliano S, aus dem Siepen M, Nattermann J, Spengler U, Lauer GM, Wiese M, Klenerman P, Bright H, Scherbaum N, Thimme R, Roggendorf M, Viazov S, Timm

J. Escape from HLA-B*08-restricted CD8 T cells by hepatitis C virus is associated with fitness costs. J Virol. 2008 Dec;82(23):11803-12. doi: 10.1128/JVI.00997-08. Epub 2008 Sep 24. PMID: 18815309; PMCID: PMC2583685.

[10] Neumann AU, Lam NP, Dahari H, Gretch DR, Wiley TE, Layden TJ, Perelson AS. Hepatitis C viral dynamics in vivo and the antiviral efficacy of interferon-alpha therapy. Science. 1998 Oct 2;282(5386):103-7. doi: 10.1126/science.282.5386.103. PMID: 9756471.

[11] Calcagnile M, Verri T, Tredici MS, Forgez P, Alifano M, Alifano P. Codon usage, phylogeny and binding energy estimation predict the evolution of SARS-CoV-2. One Health. 2021 Dec;13:100352. doi: 10.1016/j.onehlt.2021.100352. Epub 2021 Nov 24. PMID: 34841034; PMCID: PMC8610831.

[12] Hatcher EL, Zhdanov SA, Bao Y, Blinkova O, Nawrocki EP, Ostapchuck Y, Schäffer AA, Brister JR. Virus Variation Resource - improved response to emergent viral outbreaks. Nucleic Acids Res. 2017 Jan 4;45(D1):D482-D490. doi: 10.1093/nar/gkw1065. Epub 2016 Nov 28. PMID: 27899678; PMCID: PMC5210549.

[13] Vita R, Mahajan S, Overton JA, Dhanda SK, Martini S, Cantrell JR, Wheeler DK, Sette A, Peters B. The Immune Epitope Database (IEDB): 2018 update. Nucleic Acids Res. 2019 Jan 8;47(D1):D339-D343. doi: 10.1093/nar/gky1006. PMID: 30357391; PMCID: PMC6324067.

[14] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990 Oct 5;215(3):403-10. doi: 10.1016/S0022-2836(05)80360-2. PMID: 2231712.

[15] Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL. BLAST+: architecture and applications. BMC Bioinformatics. 2009 Dec 15;10:421. doi: 10.1186/1471-2105-10-421. PMID: 20003500; PMCID: PMC2803857.

[16] Viel, K. 2011. " Displaying data from NetMHCIIPan using GMAP: the SAS System as a Bioinformatics Tool." Proceedings of the PharmaSUG 2016 Conference, Denver, CO PharmaSUG. https://www.pharmasug.org/proceedings/2016/DG/PharmaSUG-2016-DG11.pdf

[17] Vita R, Overton JA, Greenbaum JA, Ponomarenko J, Clark JD, Cantrell JR, Wheeler DK, Gabbard JL, Hix D, Sette A, Peters B. The immune epitope database (IEDB) 3.0. Nucleic Acids Res. 2015 Jan;43(Database issue):D405-12. doi: 10.1093/nar/gku938. Epub 2014 Oct 9. PMID: 25300482; PMCID: PMC4384014.

[18] Vita R, Mahajan S, Overton JA, Dhanda SK, Martini S, Cantrell JR, Wheeler DK, Sette A, Peters B. The Immune Epitope Database (IEDB): 2018 update. Nucleic Acids Res. 2019 Jan 8;47(D1):D339-D343. doi: 10.1093/nar/gky1006. PMID: 30357391; PMCID: PMC6324067.

[19] Reynisson B, Alvarez B, Paul S, Peters B, Nielsen M. NetMHCpan-4.1 and NetMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data. Nucleic Acids Res. 2020 Jul 2;48(W1):W449-W454. doi: 10.1093/nar/gkaa379. PMID: 32406916; PMCID: PMC7319546.

[20] Pandey GS, Yanover C, Miller-Jenkins LM, Garfield S, Cole SA, Curran JE, Moses EK, Rydz N, Simhadri V, Kimchi-Sarfaty C, Lillicrap D, Viel KR, Przytycka TM, Pierce GF, Howard TE, Sauna ZE; PATH (Personalized Alternative Therapies for Hemophilia) Study Investigators. Endogenous factor VIII synthesis from the intron 22-inverted F8 locus may modulate the immunogenicity of replacement therapy for hemophilia A. Nat Med. 2013 Oct;19(10):1318-24. doi: 10.1038/nm.3270. Epub 2013 Sep 15. PMID: 24037092; PMCID: PMC4123441.

[21] Rambaut A, Holmes EC, O'Toole Á, Hill V, McCrone JT, Ruis C, du Plessis L, Pybus OG. A dynamic nomenclature proposal for SARS-CoV-2 lineages to assist genomic epidemiology. Nat Microbiol. 2020 Nov;5(11):1403-1407. doi: 10.1038/s41564-020-0770-5. Epub 2020 Jul 15. PMID: 32669681; PMCID: PMC7610519.

ACKNOWLEDGMENTS

The author is indebted to the NLM Support team, especially Eric Cox, Ph.D. and Cooper Park, Ph.D. who answered many questions. The author is a grateful for the charitable donation from the late Rebecca Kaneb, who also served on the board of Histonis, along with Cindy Alexander, Maria Heckscher, and Kenneth Viel.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kevin R. Viel, Ph.D.

Histonis, Incorporated
kviel@histonis.org

Navitas Data Sciences
kevin.viel@navitaslifesciences.com
www.navitasdatasciences.com

APPENDIX

Appendix 1 Suggested SAS code to read the fasta files.

```
data nucleotide_NC_045512_2
     /* protein_NC_045512_2 */
        ( drop   = __: )
     ;

  length description
         gene
         locus
         db_xref
         protein
         exception
         protein_id
         location
         gbkey          $    50
         header         $   500
         sequence       $ 32000
         ;

  if _n_ = 1
  then
    do ;
       __rcg    = prxparse( "/\[gene=([^\]]+)\]/" ) ;
       __rcl    = prxparse( "/\[locus_tag=([^\]]+)\]/" ) ;
       __rcd    = prxparse( "/\[db_xref=([^\]]+)\]/" ) ;
       __rcp    = prxparse( "/\[protein=([^\]]+)\]/" ) ;
       __rce    = prxparse( "/\[exception=([^\]]+)\]/" ) ;
       __rcpi   = prxparse( "/\[protein_id=([^\]]+)\]/" ) ;
       __rcloc  = prxparse( "/\[location=([^\]]+)\]/" ) ;
       __rcgk   = prxparse( "/\[gbkey=([^\]]+)\]/" ) ;

    end ;

  retain description
         __rcg
         __rcl
         __rcd
         __rcp
         __rce
         __rcpi
         __rcloc
         __rcgk
         gene
         locus
         db_xref
         protein
         exception
         protein_id
```

```
         location
         gbkey
         header
         sequence
         ;

  array __rc ( * )
        __rcg
        __rcl
        __rcd
        __rcp
        __rce
        __rcpi
        __rcloc
        __rcgk
        ;

  array __p ( * )
        gene
        locus
        db_xref
        protein
        exception
        protein_id
        location
        gbkey
        ;

  infile "~\nuccore_1798174254\sequence_Coding_Sequences_protein.fasta"
         /* "~\nuccore_1798174254\sequence_Coding_Sequences_protein.fasta" */
         end = end
         ;
  input ;

  if _infile_ =: ">"
  then
    do ;
       if _n_ > 1 then output ;
       header      = _infile_ ;
       call missing( of __p( * )) ;
       description = strip( scan( header , 2 , "|[")) ;
       do __i = 1 to dim( __rc ) ;
         if prxmatch( __rc( __i ) , header )
         then __p( __i ) = prxposn( __rc( __i ) , 1 , header ) ;
       end ;
       sequence    = " " ;
    end ;
   else sequence = cats( sequence
                       , _infile_
                       )  ;

  if end then output ;

run ;
```

Appendix 2 SAS Formats to translate codons to amino acids or their respective one letter symbols.

```
proc format
   library = library ;

  value $ C2AA
     "GCT"  = "Ala"
     "GCC"  = "Ala"
     "GCA"  = "Ala"
     "GCG"  = "Ala"

     "CGT"  = "Arg"
```

```
  value $ C2AAS
     "GCT"  = "A"
     "GCC"  = "A"
     "GCA"  = "A"
     "GCG"  = "A"

     "CGT"  = "R"
```

```
      "CGC"  = "Arg"
      "CGA"  = "Arg"
      "CGG"  = "Arg"
      "AGA"  = "Arg"
      "AGG"  = "Arg"

      "AAT"  = "Asn"
      "AAC"  = "Asn"

      "GAT"  = "Asp"
      "GAC"  = "Asp"

      "TGT"  = "Cys"
      "TGC"  = "Cys"

      "CAA"  = "Gln"
      "CAG"  = "Gln"

      "GAA"  = "Glu"
      "GAG"  = "Glu"

      "GGT"  = "Gly"
      "GGC"  = "Gly"
      "GGA"  = "Gly"
      "GGG"  = "Gly"

      "CAT"  = "His"
      "CAC"  = "His"

      "ATT"  = "Ile"
      "ATC"  = "Ile"
      "ATA"  = "Ile"

      "CTT"  = "Leu"
      "CTC"  = "Leu"
      "CTA"  = "Leu"
      "CTG"  = "Leu"
      "TTA"  = "Leu"
      "TTG"  = "Leu"

      "AAA"  = "Lys"
      "AAG"  = "Lys"

      "ATG"  = "Met"

      "TTT"  = "Phe"
      "TTC"  = "Phe"

      "CCT"  = "Pro"
      "CCC"  = "Pro"
      "CCA"  = "Pro"
      "CCG"  = "Pro"

      "TCT"  = "Ser"
      "TCC"  = "Ser"
      "TCA"  = "Ser"
      "TCG"  = "Ser"
      "AGT"  = "Ser"
      "AGC"  = "Ser"

      "ACT"  = "Thr"
      "ACC"  = "Thr"
      "ACA"  = "Thr"
      "ACG"  = "Thr"

      "TGG"  = "Trp"

      "TAT"  = "Tyr"
      "TAC"  = "Tyr"

      "GTT"  = "Val"
```

```
      "CGC"  = "R"
      "CGA"  = "R"
      "CGG"  = "R"
      "AGA"  = "R"
      "AGG"  = "R"

      "AAT"  = "N"
      "AAC"  = "N"

      "GAT"  = "D"
      "GAC"  = "D"

      "TGT"  = "C"
      "TGC"  = "C"

      "CAA"  = "Q"
      "CAG"  = "Q"

      "GAA"  = "E"
      "GAG"  = "E"

      "GGT"  = "G"
      "GGC"  = "G"
      "GGA"  = "G"
      "GGG"  = "G"

      "CAT"  = "H"
      "CAC"  = "H"

      "ATT"  = "I"
      "ATC"  = "I"
      "ATA"  = "I"

      "CTT"  = "L"
      "CTC"  = "L"
      "CTA"  = "L"
      "CTG"  = "L"
      "TTA"  = "L"
      "TTG"  = "L"

      "AAA"  = "K"
      "AAG"  = "K"

      "ATG"  = "M"

      "TTT"  = "F"
      "TTC"  = "F"

      "CCT"  = "P"
      "CCC"  = "P"
      "CCA"  = "P"
      "CCG"  = "P"

      "TCT"  = "S"
      "TCC"  = "S"
      "TCA"  = "S"
      "TCG"  = "S"
      "AGT"  = "S"
      "AGC"  = "S"

      "ACT"  = "T"
      "ACC"  = "T"
      "ACA"  = "T"
      "ACG"  = "T"

      "TGG"  = "W"

      "TAT"  = "Y"
      "TAC"  = "Y"

      "GTT"  = "V"
```

```
      "GTC"  = "Val"
      "GTA"  = "Val"
      "GTG"  = "Val"

      "TAA"  = "TERM"
      "TGA"  = "TERM"
      "TAG"  = "TERM"
      ;
```

```
      "GTC"  = "V"
      "GTA"  = "V"
      "GTG"  = "V"

      "TAA"  = "#"
      "TGA"  = "#"
      "TAG"  = "#"
      ;
run ;
```

Appendix 3 SAS Code to translate the nucleotide (codon) sequence to amino acids (protein) and COMPARE them.

```
data nuc_to_prot
        ( drop = AA
                 codon
                 __:
        )
      ;

  length codon        $  3
         AA           $  1
         sequence     $  32000
         ;

  set nucleotide_NC_045512_2
         ( drop   = header
                    description
           rename = ( sequence = __sequence ))
     ;

  __fl = " " ;
  do _n_ = 1 to length( __sequence ) by 3 ;
     codon = substr( __sequence , _n_ , 3 ) ;
     if length( codon ) = 3
     then
       do ;
           AA      = put( codon , $C2AAS. ) ;
           if AA ne "#"
           then sequence = cats( sequence
                                , AA
                                ) ;
            else __fl = "Y" ;
            if __fl = "Y" then put codon= AA= ;
       end ;
  end ;

run ;

proc sort
   data = nuc_to_prot ;
  by protein ;
run ;

proc sort
   data = protein_NC_045512_2
   out  = protein
   ;
  by protein ;
run ;

proc compare
   data = protein
   comp = nuc_to_prot
   ;
  id protein ;
  var sequence ;
run ;
```

Appendix 4 SAS Code to read a file, potentially with multiple fasta records, and create normalized data sets.

```
data SARS_CoV_2_id
        ( keep   = id
                   id_header
        )
     SARS_CoV_2_seq
        ( keep   = id
                   base_n
                   base
        )
     ;

  infile "~\SARS_CoV_2\sequence_Complete_Record.fasta" ;

  length id
         base_n    8
         id_header $ 500
         base      $   1
         ;

  retain id 0 ;

  input ;

  if _infile_ ~ =: ">"
  then
    do ;
       if _infile_ ne " "
       then
         do ;
            do _n_ = 1 to length( _infile_ ) ;
              base_n + 1 ;
              base = substr( _infile_ , _n_ , 1 ) ;
              if base ~ in
                    ( "A"
                    , "C"
                    , "G"
                    , "T"
                    , "N"
                    )
              then put "WARNING: "
                       _infile_
                       _n_=
                       base_n=
                       base=
                       ;
              output SARS_CoV_2_seq ;
            end ;
         end ;
    end ;
   else
     do ;
        id_header = substr( _infile_ , 2 ) ;
        id + 1 ;
        output SARS_CoV_2_id ;
     end ;
run ;
```

Appendix 5 SAS Code to generate one nucleotide substitution per reference codon and collect the resulting nonsynonymous amino acids.

```
/* Generate the nucleotide variants.
   1) At each position of a codon, substitute the other three nucleotides.
   2) Translate the resulting "variant" codon to its AA.
```

```
   3) Check if the variant and original AA do not match.
*/
data variants_NC_045512_2
        ( keep    = location
                    protein
                    AA
                    codon_original
                    codon_variant
                    AA_original
                    AA_variant
                    codon_position
                    NT_original
                    NT_variant
                    AA_m
                    NT_m
        )
     ;

  set nucleotide_NC_045512_2
        ( keep    = location
                    protein
                    sequence
        )
      ;

  array __nuc ( 4 )
       $ 1
       ( "A"
       , "C"
       , "G"
       , "T"
       ) ;

  length AA              8
         codon_original
         codon_variant   $  3
         AA_original
         AA_variant      $  1
         AA_m            $ 10
         NT_m            $ 20
         ;

  set nucleotide_NC_045512_2
         ( drop   = header
                    description
           rename = ( sequence = __sequence )
         )
    ;

  __length = length( sequence ) ;
  if mod( __length , 3 ) = 0 then __format_AA = ceil( log10( __length / 3 )) ;
   else put "W" "ARNING: "
            protein=
            __length=
            ;

  __format_NN = ceil( log10( __length )) ;

  do __AA = 1 to __length by 3 ;

     AA = ( __AA - 1 ) / 3 + 1 ;

     codon_original = substr( sequence , __AA , 3 ) ;

     if length( codon_original ) = 3
     then
       do ;

          AA_original = put( codon_original , $C2AAS. ) ;

          if     AA_original = "#"
```

```
  75              and __AA ne __length - 2
  76            then put "W" "ARNING: "
  77                     protein=
  78                     __AA=
  79                     AA_original=
  80                     ;
  81
  82            do codon_position = 1 to 3 ;
  83              NT_original = substr( codon_original , codon_position , 1 ) ;
  84              do __j = 1 to 4 ;
  85                codon_variant = codon_original ;
  86                NT_variant = __nuc( __j ) ;
  87                substr( codon_variant , codon_position , 1 ) = NT_variant ;
  88                if codon_variant ne codon_original
  89                then
  90                  do ;
  91                     AA_variant = put( codon_variant , $C2AAS. ) ;
  92                     AA_m = cats( AA_original
  93                                , "("
  94                                , strip( putn( AA
  95                                             , cats( "z"
  96                                                   , put( __format_AA , 8. )
  97                                                   , "."
  98                                                   )
  99                                             )
 100                                    )
 101                                , ")"
 102                                , AA_variant
 103                                ) ;
 104                     NT_m = cats( NT_original
 105                                , "("
 106                                , strip( putn( __AA + codon_position  - 1
 107                                             , cats( "z"
 108                                                   , put( __format_NN , 8. )
 109                                                   , "."
 110                                                   )
 111                                             )
 112                                    )
 113                                , ")"
 114                                , NT_variant
 115                                ) ;
 116                     output ;
 117                  end ;
 118              end ;
 119            end ;
 120         end ;
 121    end ;
 122
 123  run ;
```

Appendix 6 SAS Code to generate peptides of lengths 14-15 with one variant amino acid per peptide in each position of the peptide.

```
   1  proc sql ;
   2    create table distinct_variants_NC_045512_2 as
   3    select distinct location
   4         , protein
   5         , AA
   6         , AA_variant
   7    from variants_NC_045512_2
   8    where     AA_variant  ne "#"
   9          and AA_original ne AA_variant
  10    ;
  11  quit ;
  12
  13  data distinct_variants ;
  14    set distinct_variants_NC_045512_2 ;
  15    by location
```

```
16      protein
17      AA
18      ;
19    if first.AA = 0 then v_cnt + 1 ;
20     else v_cnt = 1 ;
21  run ;
22
23  proc sort
24     data = distinct_variants
25               ( drop  = location )
26     ;
27    by protein
28       v_cnt
29       AA
30       ;
31  run ;
32
33  proc datasets
34     library = WORK
35     nolist
36     ;
37    modify distinct_variants ;
38    index create vindex = ( protein
39                            v_cnt
40                            AA
41                          )
42          / nomiss
43            unique
44            ;
45  quit ;
46
47  proc sql
48     noprint ;
49    select max( v_cnt )
50              into : max_v_cnt
51    from distinct_variants
52    ;
53  quit ;
54
55  data variant_peptides_NC_045512_2
56          ( keep   = protein
57                     sequence
58                     peptide_length
59                     start
60          )
61       ;
62
63    set protein_NC_045512_2
64           ( keep   = protein
65                      sequence
66           )
67           end = end
68        ;
69
70    output ;
71
72    __length = length( sequence ) ;
73
74    __sequence = sequence ;
75
76    do peptide_length = 14 , 15 , 16 ;
77
78      do v_cnt = 1 to &max_v_cnt. ;
79
80        do start = 1 to peptide_length - 1 ;
81
82          sequence = __sequence ;
83          __fl       = " "      ;
84
85          do AA = start + 1 by peptide_length while ( AA <= __length ) ;
86
```

```
          set distinct_variants
                 key = vindex
              ;

          if _error_ = 0
          then
             do ;
                substr( sequence , AA , 1 ) =  AA_variant ;
                __fl = "Y" ;
             end ;
           else
             do ;
                _error_ = 0 ;
                _iorc_  = 0 ;
             end ;

        end ; /* CYCLED THROUGH AA */
        if __fl = "Y" then output ;
      end ; /* CYCLED THROUGH START */
    end ; /* CYCLED THROUGH V_CNT */
  end ; /* CYCLED THROUGH PEPTIDE_LENGTH */

  if end then stop ;

run ;
```

Appendix 7 SAS code to visualize multiple sequence alignments.

```
proc sql
   noprint ;
  select min( index_subject )
       , max( index_subject )
            into : min
               , : max
  from blastp_seq
  ;

  select count( distinct id )
            into : ids
  from blastp_seq
  ;
quit ;

data _null_ ;

  array seqs ( 0 : %eval( 2 * ( &ids. + 1 ) - 1 )
             , &min. : &max.
             )
        $ %eval( &max. - &min. + 1 )
        ;

  do until ( end ) ;

    set blastp_seq
           end = end
        ;

    /* Subject sequence */
    if seqs( 0
           , index_subject
           ) = " "
    then seqs( 0
           , index_subject
           ) = base_subject ;

     else if seqs( 0
                 , index_subject
                 ) ne base_subject
     then put "W" "ARNING: "
               id=
```

```
43                index=
44                seqs( 0
45                    , index_subject
46                    )=
47                base_subject=
48                ;
49
50       /* Query sequence */
51       seqs( id
52           , index_subject
53           ) = base_query ;
54
55       /* Match */
56       if prxmatch( "/^[A-Z]$/" , strip( base_match ))
57       then seqs( id + &ids.
58                , index_subject
59                ) = " " ;
60        else seqs( id + &ids.
61                  , index_subject
62                  ) = "|" ;
63
64     end ;
65
66     do __j = &min. to &max. ;
67       put seqs( 0 , __j ) @ ;
68     end ;
69     put ;
70
71     do __i = 1 to &ids. ;
72       do __j = &min. to &max. ;
73         put seqs( &ids. + __i , __j ) @ ;
74       end ;
75       put ;
76       do __j = &min. to &max. ;
77         put seqs( __i , __j ) @ ;
78       end ;
79       put ;
80     end ;
81
82   run ;
```

Appendix 8 SAS code to read output from the IEDB standalone tool mhc_II_binding.

```
1   %macro m_pgm_iedb_consensus3_read
2     ( path        =
3     , base        =
4     , delete_base = Y
5     ) ;
6
7     %let length = %sysfunc( scan( %nrbquote( &path. ) , -1 , _ )) ;
8
9     %if     &delete_base.              = Y
10        and %sysfunc( exist( &base. ))
11    %then
12       %do ;
13           proc datasets
14              library = %if %sysfunc( index( %nrbquote(&base.) , . ))
15                        %then %sysfunc( scan( %nrbquote(&base.) , 1 , . )) ;
16                         %else WORK ;
17              nolist
18              ;
19             delete %if %sysfunc( index( %nrbquote(&base.) , . ))
20                    %then %sysfunc( scan( %nrbquote(&base.) , 2 , . )) ;
21                     %else &base. ;
22                    ;
23           quit ;
24       %end ;
25
```

```
%m_u_directory_read
   ( path                 = &path.
   , out                  = dir_read_&length.
   ) ;

proc sql
   noprint ;
  select distinct pathfile
            into : pf1 -
  from dir_read_&length.
  ;
quit ;

%do __i = 1 %to &sqlobs. ;

   data __&__i.
           ( drop   = __: )
        ;

     length peptide_length                     $    2
            fn
            filename                           $  100
            allele                             $   50
            seq_num
            start
            end
            length                             8
            peptide                            $   21
            consensus_percentile_rank
            adj_consensus_percentile_rank
            comblib_core                       $   21
            comblib_score
            comblib_rank
            adjusted_comblib_rank              8
            smm_align_core                     $   21
            smm_align_ic50
            smm_align_rank
            adjusted_smm_align_rank            8
            nn_align_core                      $   21
            nn_align_ic50
            nn_align_rank
            adjusted_nn_align_rank             8
            sturniolo_core                     $   21
            sturniolo_score
            sturniolo_rank
            adjusted_sturniolo_rank            8
            ;

     infile "&&pf&__i."
            truncover
            filename  = fn
            dlm       = "09"x
            firstobs  = 2
            ;

     peptide_length = prxchange( "s/(?:.*Allele_)(\d{1,2})(?:.*)/$1/i"
                               , 1
                               , fn
                               ) ;
     filename       = scan( fn , -1 , "\" ) ;

     input allele @ ;

     if allele ne "allele" ;

     input  seq_num
            start
            end
            length
            peptide
            consensus_percentile_rank
```

```
              adj_consensus_percentile_rank

              comblib_core
              __comblib_score                  : $8.
              __comblib_rank                   : $8.
              __adjusted_comblib_rank          : $8.

              smm_align_core
              __smm_align_ic50                 : $8.
              __smm_align_rank                 : $8.
              __adjusted_smm_align_rank        : $8.
              nn_align_core
              __nn_align_ic50                  : $8.
              __nn_align_rank                  : $8.
              __adjusted_nn_align_rank         : $8.
              sturniolo_core
              __sturniolo_score                : $8.
              __sturniolo_rank                 : $8.
              __adjusted_sturniolo_rank        : $8.
              ;

        comblib_score              = input( __comblib_score            , ?? 8. ) ;
        comblib_rank               = input( __comblib_rank             , ?? 8. ) ;
        adjusted_comblib_rank      = input( __adjusted_comblib_rank    , ?? 8. ) ;
        /***/
        sturniolo_score            = input( __sturniolo_score          , ?? 8. ) ;
        sturniolo_rank             = input( __sturniolo_rank           , ?? 8. ) ;
        adjusted_sturniolo_rank    = input( __adjusted_sturniolo_rank  , ?? 8. ) ;
        /***/
        nn_align_ic50              = input( __nn_align_ic50            , ?? 8. ) ;
        nn_align_rank              = input( __nn_align_rank            , ?? 8. ) ;
        adjusted_nn_align_rank     = input( __adjusted_nn_align_rank   , ?? 8. ) ;
        /***/
        smm_align_ic50             = input( __smm_align_ic50           , ?? 8. ) ;
        smm_align_rank             = input( __smm_align_rank           , ?? 8. ) ;
        adjusted_smm_align_rank    = input( __adjusted_smm_align_rank  , ?? 8. ) ;

      run ;

      proc append
         base = &base.
         data = __&__i.
         ;
      run ;

      proc datasets
         library = WORK
         nolist
         ;
        delete __&__i. ;
      quit ;

   %end ; /* CYCLED THROUGH __i */

   %m_u_maxlen
      ( ds   = &base. )

%mend  readin ;
```