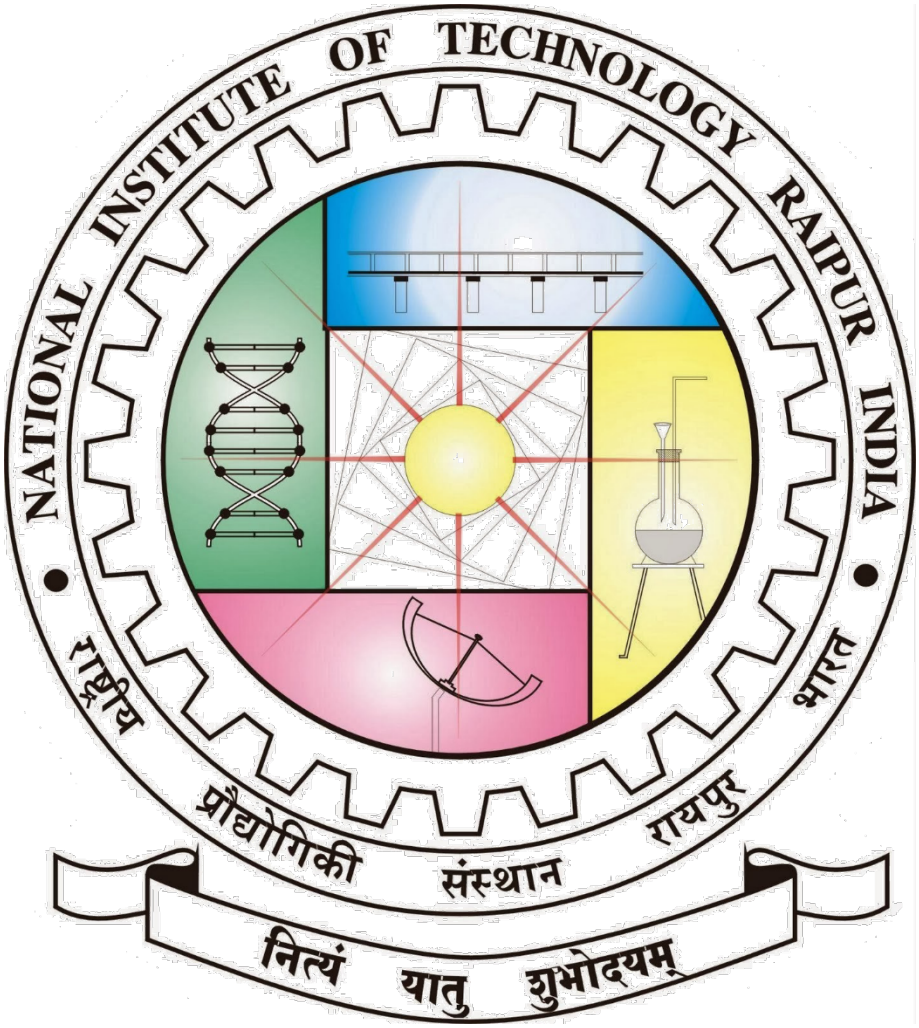
Term Paper



Under Guidance of:

Dr. Saurabh Gupta

Subject: Bioinformatics

Department of Biomedical Engineering

NIT Raipur

Submitted by:

Gaurav Kar

Roll no: 18111025

6th Semester

Analysis of FASTA, BLAST and use of Data Science for similarity searching (COVID-19 Origin)

Gaurav Kar

18111025

NIT Raipur

Abstract: Following advances in DNA and protein sequencing, the application of computational approaches in analyzing biological data has become a very important aspect of biology. Evaluating similarities between biological sequences is crucial to our understanding of evolutionary biology, and this can be achieved by basic local alignment search tool (BLAST) and fast alignment (FASTA). BLAST and FASTA have become fundamental tools of biology and it is essential to know how they operate, the task they can accomplish and how to accurately interpret their output. We will also use data science to predict from where did SARS-CoV-2 coronavirus originated from.

Introduction: I have chosen the topic "Analysis of FASTA, BLAST and use of Data Science for similarity searching (COVID-19 Origin)". I chose this topic because i was intrigued by the comparison of nucleotide or protein sequences from the same or different organisms and I wanted to know where does SARS-CoV-2 coronavirus come from? Experts think bats are the source of the virus. How did they make such a conclusion? So, in this paper I will shed light on how similarity searching algorithms and data science can be used to predict the origin of SARS-CoV-2 coronavirus from public genome data. This paper provides an analysis of BLAST and FASTA in sequence analysis. Both BLAST and FASTA algorithms are appropriate for determining highly similar sequences. However, BLAST appears to be faster and also more accurate than FASTA. Both BLAST and FASTA are limited in sensitivity and may not be able to capture highly divergent sequences in some cases. Consequently, evolutionarily diverse members of a family of proteins may be missed out in a BLAST or FASTA search. We will also see how data science in addition with FASTA & BLAST can be used to obtain better results to predict the origin of SARS-CoV-2 coronavirus.. Firstly, I will explain about similarity searching algorithm FASTA and BLAST, what was their origin and how were they immensely useful and then i will use BLAST and data science algorithms to predict the origin of coronavirus.

Main Body & Theory:

FASTA**:** The first widely-used program for database similarity searching was FASTA (Lipman and Pearson, 1985; Pearson and Lipman, 1988; Pearson, 2000). To achieve a high degree of sensitivity, this program performs optimized searches for local alignments using a substitution matrix. However, it would take a substantial amount of time to apply this strategy exhaustively.

To improve speed, the program uses the observed pattern of word hits to identify potential matches before attempting the more time-consuming optimized search. The trade-off between speed and sensitivity is controlled by the ktup parameter, which specifies the size of a word. Increasing the value of ktup decreases the number of background word hits (i.e., those that do not mark the position of an optimal alignment). This, in turn, decreases the amount of optimized searching required and improves overall search speed. The default ktup value for comparing proteins is 2, but, for finding very distant relationships, it is recommended that it be reduced to 1.

FASTA was invented in 1995 based on an improvement in FASTP, another sequence alignment tool invented in 1985 (Lipman and Pearson, 1985; Pearson, 1990). FASTP was used for protein similarity searching, however, its improvement in FASTA empowered it to execute DNA:DNA searches, translated protein:DNA searches, and also provided a more robust program for evaluating statistical significance (Pearson and Lipman, 1988, Mount, 2004). There are several different types of FASTA including TFASTAX, TFASTAY, FASTAX and FASTAY. TFASTA and TFASTAY handles query protein against a DNA library in all reading frames. FASTAX and FASTAY handle DNA query in all reading frames against a protein database. How FASTA works: Like BLAST, The FASTA program is based on a heuristic algorithm (Pearson and Lipman, 1988). FASTA sets a certain size k for k-tuple subwords (ordered set of k values). The program then searches for diagonals in the comparison matrix of the query and search sequence along which many k-tuples match. It then re-scores the highest scoring regions with the aid of a replacement matrix (a matrix that shows the rate at which a particular character in a sequence changes to other character states over time) such as the PAM250. After this, it attempts to join together the high scoring diagonals, allowing for gaps. Finally, it makes an optimal local alignment around the regions it has discovered based on the Smith-Waterman algorithm.An alternative way FASTA works is described as follows (Pearson and Lipman, 1988). In a first step FASTA tries to identify regions shared by the two sequences that have the highest density of single residue identities (ktup=1) or two-consecutive identities (ktup=2). In a second step, it then re-scans the best regions identified in the first step using the PAM-250 matrix. After that it determines if gaps can be used to join the regions identified in second step. If so, a similarity score for the gapped alignment is evaluated. Finally, it constructs an optimal alignment of the query sequence and the library sequence based on Smith-Waterman algorithm. The detailed statistical aspects involved in the FASTA algorithm have been described by Lipman and Pearson (1985). A FASTA output can be generated easily be submitting a query sequence (in a FASTA format) at a database such as UniProt. The output from FASTA is divided into four parts. The first part has some information on the database searched and the query sequence submitted; the second part is a histogram display, which reports graphically the score distribution; the third is a list of matched sequences, and related statistical information, and fourth, the alignments themselves are displayed.

The FASTA program does not investigate every word hit encountered, but instead looks initially for segments containing several nearby hits. By using a heuristic method, these segments are assigned scores and the score of the best segment found appears in the output as the init1 score. Several segments may then be combined and a new initn score is calculated from the ensemble. Most potential matches are then further evaluated by performing a search for a gapped local align- ment that is constrained to a diagonal band centered around the best initial segment. The score of this optimized alignment is shown in the output as the opt score. For those alignments finally reported (a user-specified number from the top of the hit list), a

full Smith-Waterman alignment search (without the constraining band) is performed. It should be noted that only the single optimal alignment is produced for each database sequence.

BLAST**:** BLAST is an algorithm used for comparison of amino acid sequences of different proteins or the nucleotides sequences of nucleic acid. BLAST was invented in 1990 and has since then become the defacto standard in search and alignment tools (Altschul et al., 1990). Through a BLAST search, one can compare a query sequence with a database of sequences, and thereby identify library sequences that share resemblance with the query sequence above a certain threshold. Based on such comparison, BLAST can be used to achieve several objectives including species identification, locating domains, DNA mapping and annotation (Altschul et al., 1990). There are several different types of BLAST programs available, and the choice of a BLAST programme depends on one's objective. The way most people use BLAST is to input a nucleotide or protein sequence as a query against all (or a subset of) the public sequence databases, pasting the sequence into the textbox on one of the BLAST Web pages. This sends the query over the Internet, the search is performed on the NCBI databases and servers, and the results are posted back to the person's browser in the chosen display format. However, many biotech companies, genome scientists, and bioinformatics personnel may want to use "stand-alone" BLAST to query their own, local databases or want to customize BLAST in some way to make it better suit their needs. Stand-alone BLAST comes in two forms: the executables that can be run from the command line; or the Standalone WWW BLAST Server, which allows users to set up their own in-house versions of the BLAST Web pages.

The BLAST programs introduced a number of refinements to database searching that improved overall search speed and put database searching on a firm statistical foun- dation (Altschul et al., 1990). One innovation introduced in BLAST is the idea of neighborhood words. Instead of requiring words to match exactly, a word hit is achieved if the word taken from the subject sequence has a score of at least T when a comparison is made using a substitution matrix to the word from the query. This strategy allows the word size (W) to be kept high (for speed) without sacrificing sensitivity. Thus, T becomes the critical parameter determining speed and sensitivity and W is rarely varied. If the value of T is increased, the number of background word hits will go down and the program will run faster. Reducing T allows more distant relationships to be found.

The occurrence of a word hit is followed by an attempt to find a locally optimal alignment whose score is at least equal to a score cutoff S. This is accomplished by iteratively extending the alignment both to the left and to the right, with accumulation of incremental scores for matches, mismatches, and the introduction of gaps. In practice, it is more convenient to specify an expectation cutoff E, which the program internally converts to an appropriate value of S (which would depend on the search context). In regions where matching residues are scarce, the cumulative score will begin to drop. As the mismatch and gap penalties mount, it becomes less likely that the score will rebound and ultimately reach S. This observation provides the basis for an additional heuristic whereby the extension of a hit is terminated when the reduction in score (relative to the maximum value encountered) exceeds the score dropoff threshold X. Using smaller values of X improves performance by reducing the time spent on unpromising hit extensions, at the expense of occasionally missing some true alignments.

The BLAST algorithm is a heuristic program, which means that it relies on some smart shortcuts to perform the search faster. BLAST performs "local" alignments. Most proteins are modular in nature, with functional domains often being repeated within the same protein as well as across different proteins from different species. The BLAST algorithm is tuned to find

these domains or shorter stretches of sequence similarity. The local alignment approach also means that a mRNA can be aligned with a piece of genomic DNA, as is frequently required in genome assembly and analysis. If instead BLAST started out by attempting to align two sequences over their entire lengths (known as a global alignment), fewer similarities would be detected, especially with respect to domains and motifs. When a query is submitted via one of the BLAST Web pages, the sequence, plus any other input information such as the database to be searched, word size, expect value, and so on, are fed to the algorithm on the BLAST server. BLAST works by first making a look-up table of all the "words" (short subsequences, which for proteins the default is three letters) and "neighboring words", i.e., similar words in the query sequence. The sequence database is then scanned for these "hot spots". When a match is identified, it is used to initiate gap-free and gapped extensions of the "word". BLAST does not search GenBank flatfiles (or any subset of GenBank flatfiles) directly. Rather, sequences are made into BLAST databases. Each entry is split, and two files are formed, one containing just the header information and one containing just the sequence information. These are the data that the algorithm uses. If BLAST is to be run in "stand-alone" mode, the data file could consist of local, private data, downloaded NCBI BLAST databases, or a combination of the two. Once BLAST has found a similar sequence to the query in the database, it is helpful to have some idea of whether the alignment is "good" and whether it portrays a possible biological relationship, or whether the similarity observed is attributable to chance alone. BLAST uses statistical theory to produce a bit score and expect value (E-value) for each alignment pair (query to hit). The bit score gives an indication of how good the alignment is; the higher the score, the better the alignment. In general terms, this score is calculated from a formula that takes into account the alignment of similar or identical residues, as well as any gaps introduced to align the sequences. A key element in this calculation is the "substitution matrix ", which assigns a score for aligning any possible pair of residues. The BLOSUM62 matrix is the default for most BLAST programs, the exceptions being blastn and MegaBLAST (programs that perform nucleotide–nucleotide comparisons and hence do not use protein-specific matrices). Bit scores are normalized, which means that the bit scores from different alignments can be compared, even if different scoring matrices have been used.

The E-value gives an indication of the statistical significance of a given pairwise alignment and reflects the size of the database and the scoring system used. The lower the E-value, the more significant than the hit. A sequence alignment that has an E-value of 0.05 means that this similarity has a 5 in 100 (1 in 20) chance of occurring by chance alone. Although a statistician might consider this to be significant, it still may not represent a biologically meaningful result, and analysis of the alignments is required to determine "biological" significance.

There are several variants of BLAST, each distinguished by the type of sequence (DNA or protein) of the query and database sequences. The BLASTP program compares a protein query to a protein database. The corresponding program for nucleotide sequences is BLASTN. If the sequence types differ, the DNA sequence can be translated by the program (in all six reading frames) and compared to the protein sequence. BLASTX compares a DNA query sequence to the protein database, which is useful for analyzing new sequence data and ESTs. For a protein query against a nucleotide database, use the TBLASTN program. This is useful for finding unannotated coding regions in database sequences.

Table 1. BLAST programmes that are commonly used.

Program	Query sequence type	Target sequence type
BLASTP	Protein	Protein
BLASTN	Nucleotide	Nucleotide
BLASTX	Nucleotide (translated)	Protein
TBLASTN	Protein	Nucleotide (translated)
TBLASTX	Nucleotide (translated)	Nucleotide (translated)

BLAST vs. FASTA in terms of precision, accuracy and algorithm runtime complexity: BLAST and FASTA are similar in that both programs are based on a common assumption that true matches are likely to have at least some short stretches of high-scoring similarity, but whereas FASTA targets exactly matching 'words' (strings of residues), BLAST employs a scoring matrix - BLOSUM62 for amino-acid sequences.

It is worthwhile defining the terminologies precision, accuracy and algorithm runtime complexity in the context of bioinformatics software applications, as they form the basis of comparison of BLAST and FASTA in this section. Algorithm runtime complexity refers to how fast the bioinformatics tool performs and produces a results output. Accuracy is the degree of closeness of a measurement to its actual or true value. The precision of a measurement system is the extent to which repeated measurements under unvaried conditions yield the same results. It is important to note that accuracy of a bioinformatics tool can be evaluated by its sensitivity (ability to identify or recognize distantly related sequences), and then accuracy and precision may be used interchangeably. In terms of algorithm runtime complexity, BLAST is faster than FASTA by searching for only the more significant patterns in the sequences. The sensitivity (or accuracy) of BLAST and FASTA tends to be different for nucleic acid and protein sequences.

For protein alignments, the BLAST heuristic algorithm is more sensitive than that of FASTA, although BLAST employs a word size of three for proteins while FASTA works with a word size of two. In the case of nucleic acid sequences, BLAST employs a long word size of eleven. However, the heuristic algorithm that improves the sensitivity for protein sequences does not work as well for nucleic acid (Li et al., 2004), and FASTA is more sensitive than BLAST for nucleic acid sequences. BLAST and FAST A differ is the statistical evaluation of their output which would most likely affect their relative accuracy. FAST A produces an E-value that shows relatively accurate estimation for found matches and the expectancy to find them by chance (Brenner et al., 1998). On the other hand, BLAST calculates expectancy by removing the results scored lower than the threshold value (Brenner et al., 1998; Sansom, 2000). This results in elimination of statistically not significant alignments increasing the accuracy of BLAST over FASTA (Sansom, 2000). Despite the high numbers of citations of BLAST and FAST A in literature, there is hardly any quantitative comparison of the two tools in terms of speed, precision and accuracy. A comparative analysis on the algorithm runtime complexity and precision of BLAST and FASTA showed that BLAST was over six times faster for searching structural classification of proteins (SCOP) than FASTA (Chattaraj et al., 1999). However, the average precision of FASTA was about 2°/o higher than that of BLAST (Chattaraj et al., 1999). In the same study, it was also observed that Psi-BLAST is almost an order of magnitude slower

than BLAST but over 3% more accurate in average precision (Chattaraj et al., 1999). It is important to note that both FASTA and BLAST allow gaps at some point in their mechanism of operation and for this reason, both methods have the potential to miss significant similarities present in the database. Another setback of both search tools is that many proteins are multifunctional multi-domain proteins, and thus a high hit to one domain does not necessarily define function.

Uses of BLAST:

Identifying species**:** With the use of BLAST, you can possibly correctly identify a species or find homologous species. This can be useful, for example, when you are working with a DNA sequence from an unknown species**.**

Locating domains**:** When working with a protein sequence you can input it into BLAST, to locate known domains within the sequence of interest.

Establishing phylogeny**:** Using the results received through BLAST you can create a phylogenetic tree using the BLAST web-page. Phylogenies based on BLAST alone are less reliable than other purpose-built computational phylogenetic methods, so should only be relied upon for "first pass" phylogenetic analyses.

DNA mapping**:** When working with a known species, and looking to sequence a gene at an unknown location, BLAST can compare the chromosomal position of the sequence of interest, to relevant sequences in the database(s). NCBI has a "Magic-BLAST" tool built around BLAST for this purpose.

Comparison**:** When working with genes, BLAST can locate common genes in two related species, and can be used to map annotations from one organism to another.

Now that we have explained FASTA and BLAST, how it works and what are its applications, we can study our topic “Where does SARS-CoV-2 come from”. We’ll use BLAST and Data Science to research more about this topic and form our conclusion from it.

First, we find the genome of SARS-CoV-2, there are hundreds of complete sequences of SARS-CoV-2 available on the GenBank. For example, we are choosing the Wuhan variant LR757998. Then, we will search LR757998 on BLASTN.

After searching on BLASTN we find out that there is plethora of complete sequences similar to the Wuhan LR757998, after scrolling a bit we find out a genome sequence which is 96% similar to the Wuhan variant named MN996532, which is related to Bats.

So basically, Here I show how to find the similarity between current coronavirus SARS-Cov-2 to the one from bats. I use the genome sequence of SARS-Cov-2 and search for similar sequences in BLASTN. I apply some GNU/Linux bash commands to filter results.

GenBank

Severe acute respiratory syndrome coronavirus 2 genome assembly, chromosome: whole_genome

GenBank: LR757998.1

FASTA Graphics

Go to:

```
LOCUS       LR757998              29866 bp    RNA     linear   VRL 25-SEP-2020
DEFINITION  Severe acute respiratory syndrome coronavirus 2 genome assembly,
            chromosome: whole_genome.
ACCESSION   LR757998
VERSION     LR757998.1
DBLINK      BioProject: PRJEB36487
            BioSample: SAMEA6507890
KEYWORDS    .
SOURCE      Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2)
  ORGANISM  Severe acute respiratory syndrome coronavirus 2
            Viruses; Riboviria; Orthornavirae; Pisuviricota; Pisoniviricetes;
            Nidovirales; Cornidovirineae; Coronaviridae; Orthocoronavirinae;
            Betacoronavirus; Sarbecovirus.
REFERENCE   1
  AUTHORS   Hunter,C. and Wei,X.
  TITLE     Direct Submission
  JOURNAL   Submitted (30-JAN-2020) BGI, GigaScience, 16 Dai Fu Street, Tai Po
            Industrial Estate, N.T., Hong Kong
COMMENT     Coronovirus sequence assembly isolated from 2019/2020 Wuhan
            outbreak patient.
FEATURES             Location/Qualifiers
     source          1..29866
                     /organism="Severe acute respiratory syndrome coronavirus
                     2"
                     /mol_type="genomic RNA"
                     /host="Homo sapiens"
                     /db_xref="taxon:2697049"
                     /chromosome="whole_genome"
                     /country="China:Wuhan"
                     /collection_date="2019-12-26"
```

Sequences producing significant alignments Download Select columns Show 100

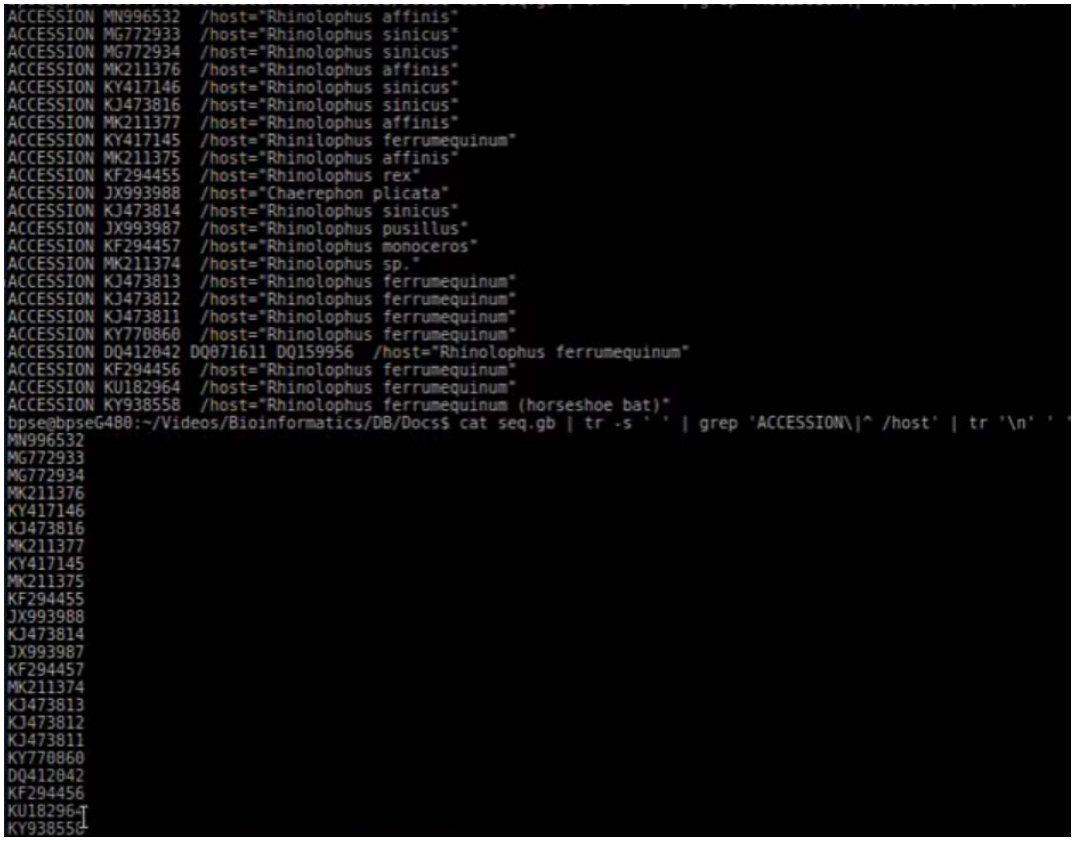
select all *100 sequences selected* GenBank Graphics Distance tree of results MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
Severe acute respiratory syndrome coronavirus 2 genome assembly, chromosome: whole_genome	Severe acute re...	55153	55153	100%	0.0	100.00%	29866	LR757998.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/CA-CZB-1763/2020, com...	Severe acute re...	55142	55142	100%	0.0	99.99%	29901	MT671817.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/CA-CZB-1209/2020, com...	Severe acute re...	55142	55142	100%	0.0	99.99%	29893	MT499184.1
Synthetic construct ORF1ab, spike, ORF3, E, M, ORF6, ORF8, and N genes, complete cds	synthetic construct	55142	55142	100%	0.0	99.99%	29891	MT108784.1
Severe acute respiratory syndrome coronavirus 2 isolate WIV04, complete genome	Severe acute re...	55142	55142	100%	0.0	99.99%	29891	MN996528.1
Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome	Severe acute re...	55142	55142	100%	0.0	99.99%	29903	NC_045512.2
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/UT-UPHL-2102921286/2...	Severe acute re...	55142	55142	100%	0.0	99.99%	29904	MW566244.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/UT-UPHL-2102342783/2...	Severe acute re...	55142	55142	100%	0.0	99.99%	29904	MW562722.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/KOR/KCDC12-NCCP43330/2...	Severe acute re...	55142	55142	100%	0.0	99.99%	29900	MW466795.1
Severe acute respiratory syndrome coronavirus 2 isolate BetaCoV/Wuhan/IPBCAMS-WH-04/2019, complete ...	Severe acute re...	55140	55140	99%	0.0	99.99%	29890	MT019532.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/CA-CZB-1757/2020, com...	Severe acute re...	55136	55136	100%	0.0	99.99%	29890	MT671813.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/CA-CZB-1756/2020, com...	Severe acute re...	55136	55136	100%	0.0	99.99%	29892	MT671812.1
Severe acute respiratory syndrome coronavirus 2 TKYE6938_2020 RNA, complete genome	Severe acute re...	55136	55136	100%	0.0	99.99%	29889	LC553269.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/CA-CZB-1226/2020, com...	Severe acute re...	55136	55136	100%	0.0	99.99%	29902	MT499201.1
Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/CA-CDC-0139/2020, co...	Severe acute re...	55136	55136	100%	0.0	99.99%	29903	MT481992.1

select all *100 sequences selected* GenBank Graphics Distance tree of results MSA Viewer

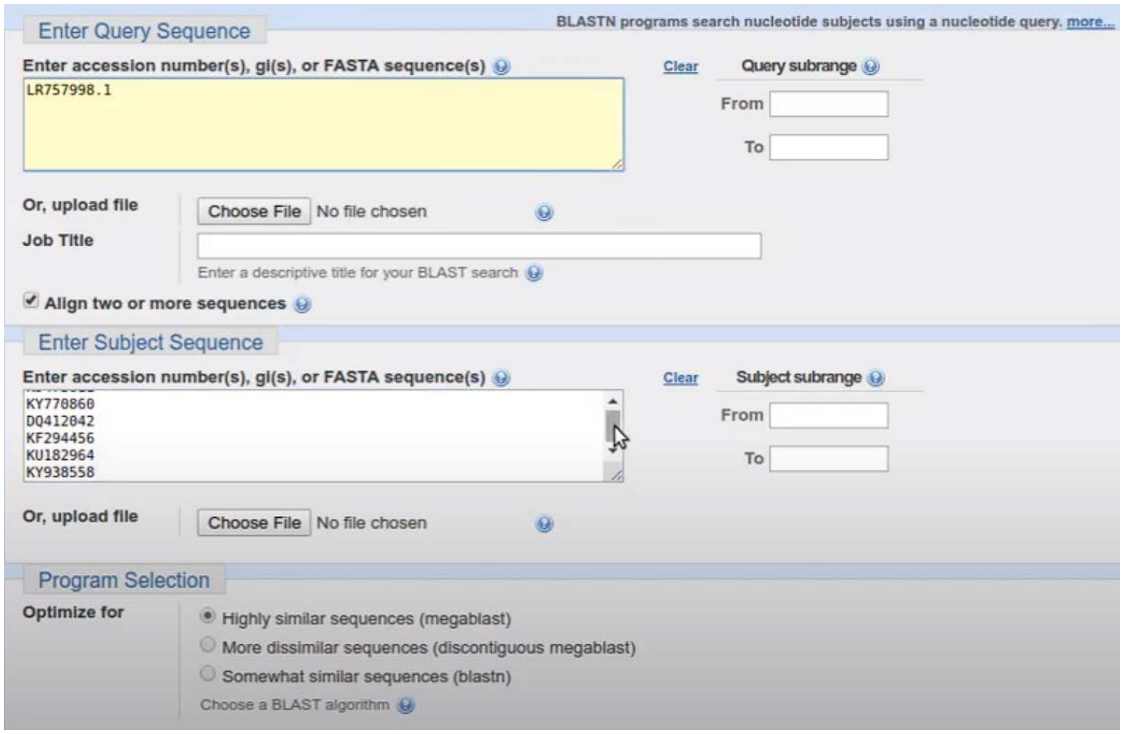
Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
Synthetic construct ORF1ab, spike, ORF3, E, M, ORF6, ORF8, and N genes, complete cds	synthetic construct	55142	55142	100%	0.0	99.99%	29891	MT108784.1
Synthetic construct clone icSARS-CoV-2-WT ORF1ab polyprotein (ORF1ab), ORF1a polyprotein (ORF1ab), s...	synthetic construct	55119	55119	100%	0.0	99.98%	29903	MT461669.1
Synthetic construct clone icSARS-CoV-2-nLuc-GFP ORF1ab polyprotein (ORF1ab), ORF1a polyprotein (ORF1...	synthetic construct	50579	54611	99%	0.0	99.98%	30857	MT461671.1
Synthetic construct clone icSARS-CoV-2-GFP ORF1ab polyprotein (ORF1ab), ORF1a polyprotein (ORF1ab), s...	synthetic construct	50579	54611	99%	0.0	99.98%	30347	MT461670.1
Bat coronavirus RaTG13, complete genome	Bat coronavirus ...	48706	48706	99%	0.0	96.13%	29855	MN996532.2
Pangolin coronavirus isolate MP789, complete genome	Pangolin corona...	32154	38686	98%	0.0	90.10%	29521	MT121216.1
Pangolin coronavirus isolate PCoV_GX-P5L, complete genome	Pangolin corona...	28301	31367	99%	0.0	85.98%	29806	MT040335.1
Pangolin coronavirus isolate PCoV_GX-P4L, complete genome	Pangolin corona...	28293	31354	99%	0.0	85.97%	29805	MT040333.1
Pangolin coronavirus isolate PCoV_GX-P2V, complete genome	Pangolin corona...	28262	31323	99%	0.0	85.94%	29795	MT072864.1
Pangolin coronavirus isolate PCoV_GX-P1E, complete genome	Pangolin corona...	28256	31323	99%	0.0	85.95%	29801	MT040334.1
Pangolin coronavirus isolate PCoV_GX-P5E, complete genome	Pangolin corona...	28247	31302	99%	0.0	85.95%	29802	MT040336.1
Bat coronavirus RacCS264, partial genome	Bat coronavirus ...	27464	42006	92%	0.0	94.69%	29820	MW251311.1

Now we apply filtering by using bash commands:



So, after filtering we get list of variants of virus which are similar to SARS-CoV-2 but the hosts are Bats.

And Now we compare the Wuhan variant which is base variant to the filtered ones using BLASTN again.



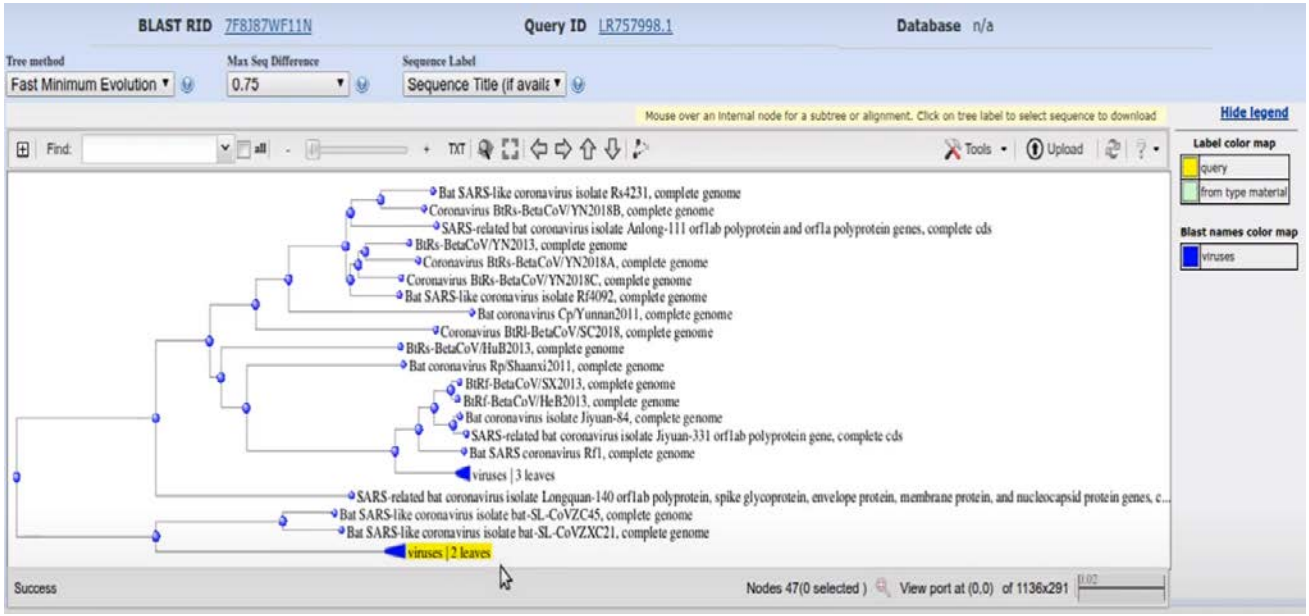
And the results are:

Description	Max Score	Total Score	Query Cover	E value	Per. Ident	Accession
Bat coronavirus RaTG13, complete genome	48700	48700	99%	0.0	96.11%	MN996532.1
Bat SARS-like coronavirus isolate bat-SL-CoVZC45, complete genome	26906	35284	95%	0.0	89.11%	MG772933.1
Bat SARS-like coronavirus isolate bat-SL-CoVZXC21, complete genome	22218	35224	94%	0.0	88.65%	MG772934.1
Bat SARS-like coronavirus isolate Rs4231, complete genome	15171	22512	91%	0.0	82.30%	KY417146.1
Coronavirus BtRs-BetaCoV/YN2018B, complete genome	15171	22596	91%	0.0	82.31%	MK211376.1
BtRs-BetaCoV/YN2013, complete genome	15158	21667	85%	0.0	82.28%	KJ473816.1
Coronavirus BtRs-BetaCoV/YN2018C, complete genome	15143	22381	88%	0.0	82.29%	MK211377.1
Bat SARS-like coronavirus isolate Rf4092, complete genome	15128	22350	87%	0.0	82.26%	KY417145.1
Coronavirus BtRs-BetaCoV/YN2018A, complete genome	15112	22349	89%	0.0	82.25%	MK211375.1
SARS-related bat coronavirus isolate Anlong-111 orf1ab polyprotein and orf1a polyprotein genes, complete cds	15079	16310	64%	0.0	82.35%	KF294455.1
Bat coronavirus Cp/Yunnan2011, complete genome	15038	21828	86%	0.0	82.17%	JX993988.1
BtRs-BetaCoV/HuB2013, complete genome	14964	22334	87%	0.0	82.20%	KJ473814.1
Bat coronavirus Rp/Shaanxi2011, complete genome	14886	21814	88%	0.0	82.00%	JX993987.1
SARS-related bat coronavirus isolate Longquan-140 orf1ab polyprotein, spike glycoprotein, envelope protein, membrane protein, and nucleocap	14754	23743	91%	0.0	81.88%	KF294457.1
Coronavirus BtRl-BetaCoV/SC2018, complete genome	14726	21812	87%	0.0	82.01%	MK211374.1
BtRf-BetaCoV/SX2013, complete genome	14717	21355	87%	0.0	81.82%	KJ473813.1

All of these above viruses' hosts are bats and related to our base variant Wuhan LR757998.

So, we can conclude that SARS-CoV-2 originated from bats.

For more information we can also look at Distance trees:



Previously, we found the origin of coronavirus by using traditional bioinformatics method BLAST sequence alignment tool, now we will see how can we get the same result by using data science.

Using Data Science to Predict the Origin of SARS-CoV-2 Coronavirus from Public Genome Data

Traditionally, most medical scientists will use traditional bioinformatics tools such as BLAST sequence alignments. However, in this term paper, I will show how we can analyse the coronavirus genomes from a data science perspective. Specifically, we will use data science methods to predict the origin of the COVID-19 by comparing its genome sequences with other

coronaviruses that have been infected on other animals such as Chicken, Cattle, Duck, and Bats. Based on our analysis, we will demonstrate how bats are shown to have the highest genome sequence similarity when compared to the other animals.

Datasets

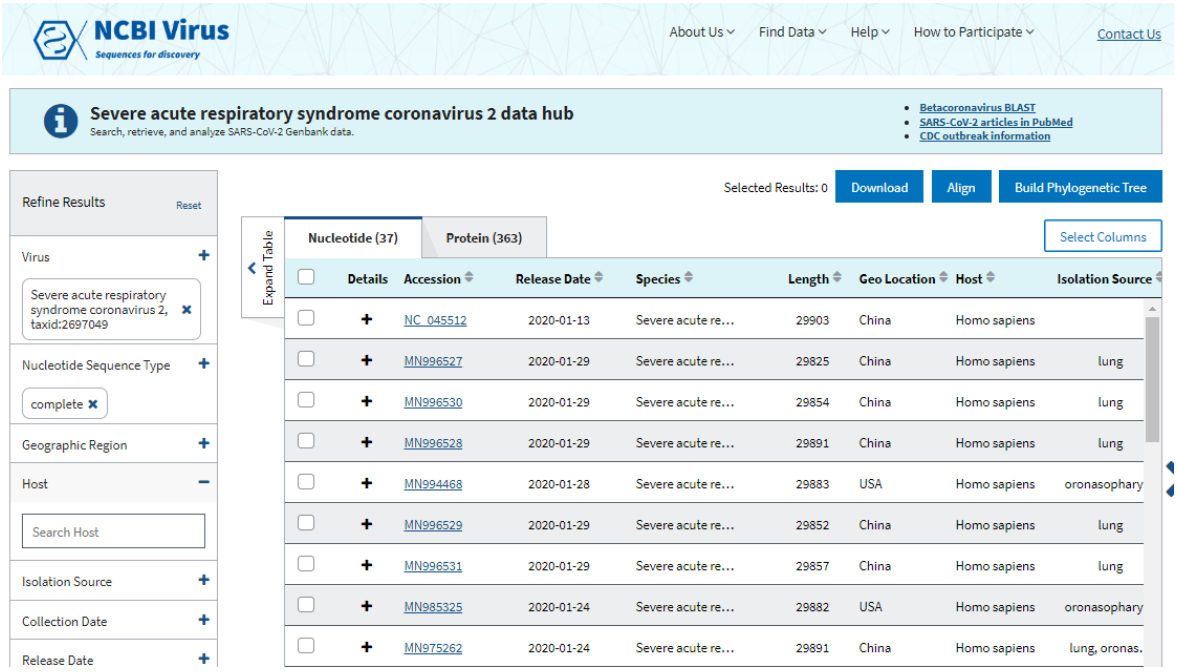
The genome sequence of the 2019 Coronavirus is made available to the public via the U.S National Library of Medicine website (https://www.ncbi.nlm.nih.gov/labs/virus). We will notice that there are 2 types of main types of viral genome sequences: nucleotide and protein. Here, we will specifically analyse the nucleotide sequences of 4 animals, namely, chicken, bat, cattle, and duck. We will compare them with the 2019 coronavirus nucleotide sequences collected from infected people. We can also use the same methods learned here to analyse the protein sequences as well.

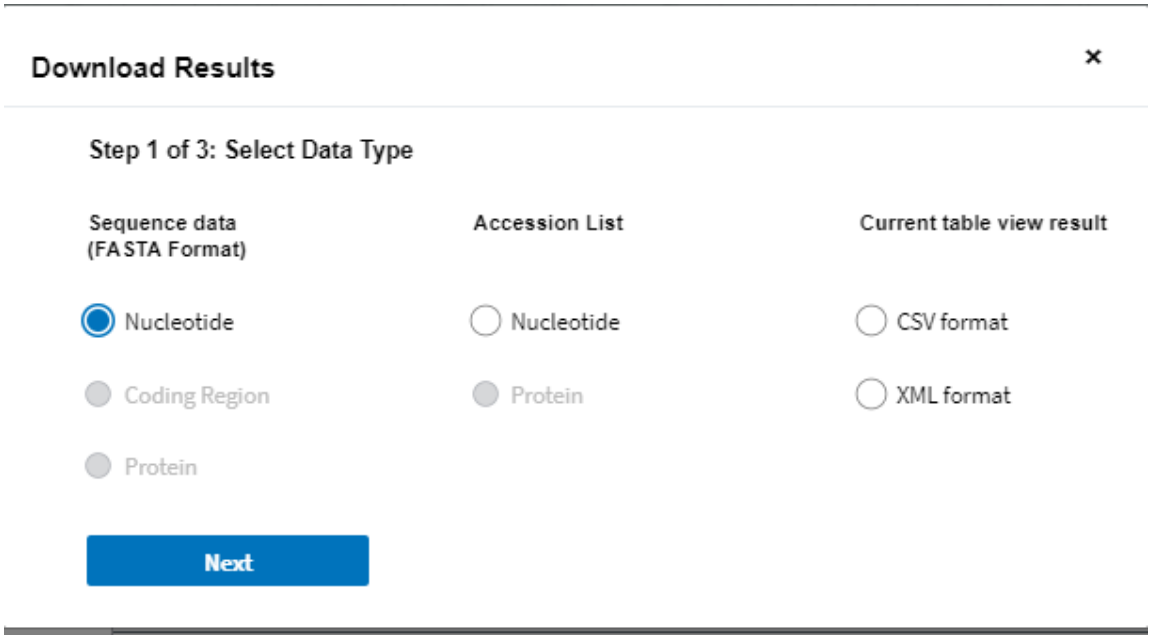
Data Collection

The main goal of this analysis is to determine the likely origin of the virus. In practice, we will need to scan the COVID-19 coronavirus genome with all viruses that circulate among animals. However, due to the limited computing resources, we will only analyse coronaviruses that circulate from chicken, bat, cattle, and duck only.

First, let's download the nucleotide sequences of the COVID-19 coronavirus by clicking the nucleotide link from the NCBI virus web page.

The web page is self-exploratory to use. We can observe that the "Severe acute respiratory syndrome coronavirus 2" is automatically selected under the Virus section of the Refine Results Filter. Further, there are two different sequence types: *complete* or *refseq*. For this walkthrough, choose complete sequence type from the nucleotide sequence type section. Next, click on the Download blue button at the top right of the page. The genome sequence will be downloaded to our machine as. fasta file format.





Once I've downloaded the sequence data, let's open it on your favourite text editor to see what the genome sequence looks like.

```
>MN908947 |Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1| complete genome
ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCT
GTTCTCTAAACGAACTTTAAAATCTGTGTGGCTGTCACTCGGCTGCATGCTTAGTGCACT
CACGCAGTATAATTAATAACTAATTACTGTCGTTGACAGGACACGAGTAACTCGTCTATC
TTCTGCAGGCTGCTTACGGTTTCGTCCGTGTTGCAGCCGATCATCAGCACATCTAGGTTT
CGTCCGGGTGTGACCGAAAGGTAAGATGGAGAGCCTTGTCCCTGGTTTCAACGAGAAAAC
ACACGTCCAACTCAGTTTGCCTGTTTTACAGGTTCGCGACGTGCTCGTACGTGGCTTTGG
AGACTCCGTGGAGGAGGTCTTATCAGAGGCACGTCAACATCTTAAAGATGGCACTTGTGG
CTTAGTAGAAGTTGAAAAAGGCGTTTTGCCTCAACTTGAACAGCCCTATGTGTTCATCAA
ACGTTCGGATGCTCGAACTGCACCTCATGGTCATGTTATGGTTGAGCTGGTAGCAGAACT
CGAAGGCATTCAGTACGGTCGTAGTGGTGAGACACTTGGTGTCCTTGTCCCTCATGTGGG
CGAAATACCAGTGGCTTACCGCAAGGTTCTTCTTCGTAAGAACGGTAATAAAGGAGCTGG
TGGCCATAGTTACGGCGCCGATCTAAAGTCATTTGACTTAGGCGACGAGCTTGGCACTGA
TCCTTATGAAGATTTTCAAGAAAACTGGAACACTAAACATAGCAGTGGTGTTACCCGTGA
ACTCATGCGTGAGCTTAACGGAGGGGCATACACTCGCTATGTCGATAACAACTTCTGTGG
CCCTGATGGCTACCCTCTTGAGTGCATTAAAGACCTTCTAGCACGTGCTGGTAAAGCTTC
ATGCACTTTGTCCGAACAACTGGACTTTATTGACACTAAGAGGGGTGTATACTGCTGCCG
TGAACATGAGCATGAAATTGCTTGGTACACGGAACGTTCTGAAAAGAGCTATGAATTGCA
GACACCTTTTGAAATTAAATTGGCAAAGAAATTTGACACCTTCAATGGGGAATGTCCAAA
TTTTGTATTTCCCTTAAATTCCATAATCAAGACTATTCAACCAAGGGTTGAAAAGAAAAA
GCTTGATGGCTTTATGGGTAGAATTCGATCTGTCTATCCAGTTGCGTCACCAAATGAATG
CAACCAAATGTGCCTTTCAACTCTCATGAAGTGTGATCATTGTGGTGAAACTTCATGGCA
GACGGGCGATTTTGTTAAAGCCACTTGCGAATTTTGTGGCACTGAGAATTTGACTAAAGA
AGGTGCCACTACTTGTGGTTACTTACCCCAAAATGCTGTTGTTAAAATTTATTGTCCAGC
ATGTCACAATTCAGAAGTAGGACCTGAGCATAGTCTTGCCGAATACCATAATGAATCTGG
CTTGAAAACCATTCTTCGTAAGGGTGGTCGCACTATTGCCTTTGGAGGCTGTGTGTTCTC
TTATGTTGGTTGCCATAACAAGTGTGCCTATTGGGTTCCACGTGCTAGCGCTAACATAGG
TTGTAACCATACAGGTGTTGTTGGAGAAGGTTCCGAAGGTCTTAATGACAACCTTCTTGA
AATACTCCAAAAAGAGAAAGTCAACATCAATATTGTTGGTGACTTTAAACTTAATGAAGA
GATCGCCATTATTTTGGCATCTTTTTCTGCTTCCACAAGTGCTTTTGTGGAAACTGTGAA
```

We can observe that the first line starts with '>' that denotes a single-line description of the virus. At this point of writing, there are 34 COVID-19 coronavirus samples that have been made available at the NCBI virus database.

Next, let's download the coronavirus nucleotide sequences of the four animals (i.e., chicken, cattle, duck, and bat).

To do this, remove the COVID-19 coronavirus option (i.e. Severe acute respiratory syndrome coronavirus 2) from the Virus section of the Refine Results, and search for Coronaviradae instead. Coronaviradae is the scientific name for all coronaviruses that circulate among animals. Selecting this option will give you a list of nucleotide sequence samples of all types of coronavirus family that have been collected by the medical scientists in the past. Interestingly, you can see from the search result that the first collected genome sample of coronavirus was collected in 1983!

You may start by collecting the chicken nucleotide genome sequence first. The scientific name for chicken is *gallus gallus*. Proceed to the Host section at the Refine Results, and search for Gallus gallus (chicken). Once selecting the option, the search result list will be updated and you may click on the Download button once again to download the complete nucleotide sequence of the chicken.

Repeat these steps for the remaining 3 animals: 1) cattle (bos taurus), 2) duck (anatidae), and 3) bat (chiropter).

Data Transformation

Our first task is to read the COVID-19 and the other four animal genome sequences from the sequence data fasta files and place them onto the pandas data frame. In achieving this, we will need to extract only the sequence characters (e.g., *ATTAAAG*) and ignore all the single-line description which begins with '>'. The following code identifies each nucleotide sequence by '>' and extracts the sequence characters in between two '>' symbols. The extracted sequence characters are then appended sequentially into a list. The list contains the whole string for each animal genome.

```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
def process_file(filename,target_val):
    f = open(filename) #'datasets\\corona-nucleo-chicken-complete.fasta')
    lines = ""
    s1 = list()
    step = 0
    term = 0
for line in f:
        line = ''.join(line.split())
        if line.startswith(">") and step==0:
            line = line.split('>',1)[0].strip()
```

```
        step = step + 1
if line.startswith(">") and step>=1: #and step>=1:
        line = line.split('>',1)[0].strip()
        s1.append(lines)
        lines = ""
        step = step + 1
        term = 0
    lines = lines + line
```

Numeric Representation of the Sequences

Next, we will need to generate features from the strings that contain our sequence characters. Since the sequence is represented as a series of alphabetical characters, we will need to split the sequence characters into small tokens. For example, a token with 2 characters only, a token with 3 characters, 4 characters only, etc. Furthermore, we will need to convert the tokens into a numerical representation format so that we can feed them into the machine learning algorithm for prediction. One common way is to apply the bag-of-words data representation that computes the frequency/occurrences of each token. It keeps track of two important information: A vocabulary of sequence characters and the number of occurrences for each token.

To apply both n-grams and bag-of-words together, we can use the CountVectorizer API from the scikit-learn machine learning library.

```
count_vect = CountVectorizer(lowercase=False, ngram_range=(2,4),analyzer='char')
 X1 = count_vect.fit_transform(s1)
```

For this walkthrough, we will only set the minimum n-grams to start from 2 to 4 sequence of tokens. For example, 2-gram has at most 2 characters (e.g., AT), 3-gram (e.g., ACG), and 4-gram (e.g., ACTT).

```
def generate_ngrams(s1):
   count_vect = CountVectorizer(lowercase=False, ngram_range=(2,4),analyzer='char')
   X1 = count_vect.fit_transform(s1)

   lcount = list()
   lcount = []
   for i in s1:
      count = len(i)
```

```
        #print(count)
        lcount.append(count)
count_vect_df = pd.DataFrame(X1.todense(), columns=count_vect.get_feature_names())

    count_vect_df=count_vect_df.apply(lambda x: x / lcount[x.name] ,axis=1)

    return count_vect_df
```

Data Labelling

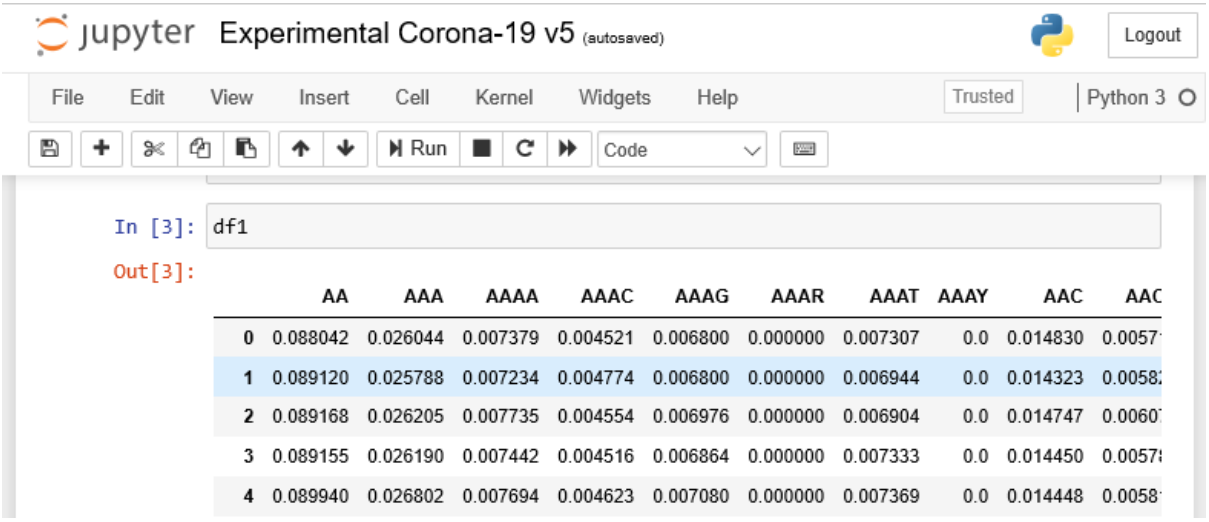
Once the data has been transformed into numerical values as features, we will need to label them for prediction.

```
df1 = process_file('datasets\\corona-seq-chicken.fasta',"chicken")
df2 = process_file('datasets\\corona-seq-duck.fasta',"duck")
df3 = process_file('datasets\\corona-seq-cattle.fasta',"cattle")
df4 = process_file('datasets\\corona-seq-bat.fasta',"bat")
```

Here we can see that each fasta sequence file is assigned with its own label (i.e., chicken, duck, cattle, and bats). The labeling is done at the end of the data transformation process.

```
def generate_ngrams(s1):
.....
count_vect_df = generate_ngrams(s1)
count_vect_df['target'] = target_val
return count_vect_df
```

If we've done everything correctly, we should be able to view all the features as numerical values in pandas data frame format.



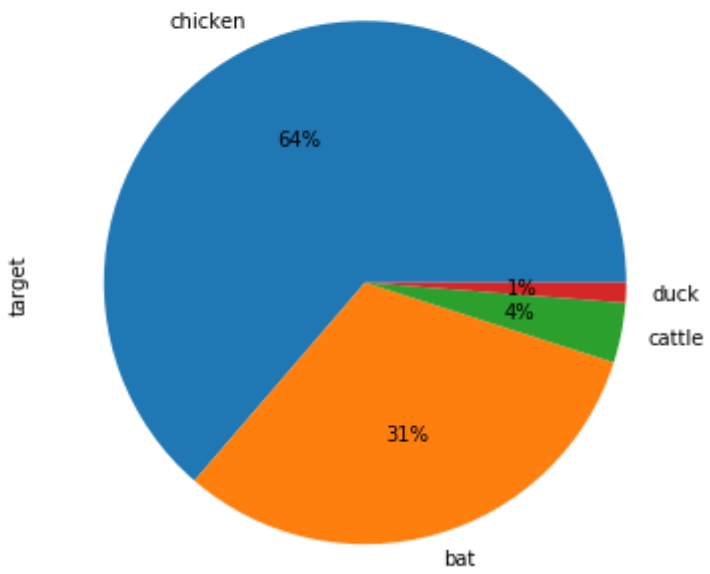
	AA	AAA	AAAA	AAAC	AAAG	AAAR	AAAT	AAAY	AAC	AAC
0	0.088042	0.026044	0.007379	0.004521	0.006800	0.000000	0.007307	0.0	0.014830	0.0057
1	0.089120	0.025788	0.007234	0.004774	0.006800	0.000000	0.006944	0.0	0.014323	0.0058
2	0.089168	0.026205	0.007735	0.004554	0.006976	0.000000	0.006904	0.0	0.014747	0.0060
3	0.089155	0.026190	0.007442	0.004516	0.006864	0.000000	0.007333	0.0	0.014450	0.0057
4	0.089940	0.026802	0.007694	0.004623	0.007080	0.000000	0.007369	0.0	0.014448	0.0058

Exploratory Analysis

Having all the sequences represented in data frame format, we can explore the % distribution of our coronavirus datasets. We can observe that 64% of our datasets are nucleotide sequences of the chicken, 31% from the bats, and the remaining 4% and 1% from the cattle and duck, respectively.

```
import matplotlib.pyplot as plt

plot_size = plt.rcParams["figure.figsize"]

plot_size[0] = 8

plot_size[1] = 6

plt.rcParams["figure.figsize"] = plot_size

df7['target'].value_counts().plot(kind='pie', autopct='%1.0f%%')
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x15c88eb0c50>
```



Building the Predictive Model

We are now ready to build the predictive model by applying our labelled nucleotide datasets to the machine learning algorithm. In this case, we are going to use the XGBoost Gradient Boosting algorithm to build our predictive model.

```
from sklearn.model_selection import train_test_split

from xgboost import XGBClassifier

from xgboost import plot_importance

import xgboost

# create a train/test split
```

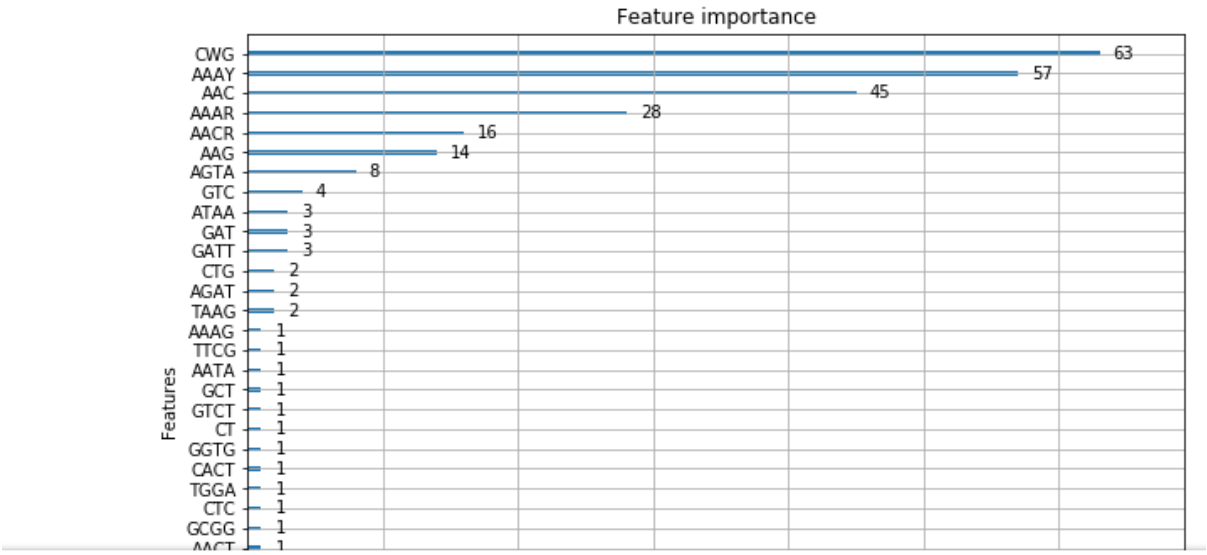
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7, shuffle=True)

model = XGBClassifier()

model.fit(X_train, y_train)

Using the model, we built, we can further analyse to determine which regions of the nucleotide sequences have the most influential factors in determining which sequences belong to which animals. We can observe that “CWG”, “AAAY”, and “AAC” are the top 3 sequence regions that differentiate the genomes of the 4 animals.

```
In [25]: import matplotlib.pyplot as plt

         ax1 = plot_importance(model)
         ax1.figure.set_size_inches(10,8)
```



Data Transformation and Text Representation for the COVID-19 coronavirus

Next, we need to load the COVID-19 nucleotide sequence file to our program in order to do prediction.

Similar to the animal coronaviruses sequences, we are required to perform data transformation and numeric representation for the COVID-19 coronavirus. Both n-gram and Bag-of-words algorithms will need to be applied but additional steps are also necessary. We will explain these steps in detail.

First, we will load the nucleotide of the COVID-19 coronavirus from our Fasta file

cov = process_file(‘datasets\\corona-covid-19.fasta’,“COVID-19”)

cov

	AA	AAA	AAAA	AAAC	AAAG	AAAT	AAC	AACA	AACC	AACG	...	YC	YCA
0	0.096311	0.030866	0.009397	0.006387	0.006855	0.008193	0.020566	0.008561	0.003846	0.001505	...	0.000000	0.000000
1	0.095281	0.029828	0.008412	0.006368	0.006837	0.008211	0.020544	0.008546	0.003821	0.001508	...	0.000000	0.000000
2	0.095676	0.030185	0.008701	0.006392	0.006860	0.008199	0.020581	0.008567	0.003848	0.001506	...	0.000000	0.000000
3	0.095710	0.030185	0.008701	0.006392	0.006860	0.008199	0.020581	0.008534	0.003848	0.001539	...	0.000000	0.000000
4	0.095707	0.030218	0.008734	0.006392	0.006860	0.008199	0.020547	0.008533	0.003848	0.001506	...	0.000000	0.000000
5	0.095676	0.030185	0.008701	0.006392	0.006860	0.008199	0.020581	0.008567	0.003848	0.001506	...	0.000000	0.000000

Since we are going to do the prediction (instead of building model), we can drop the target label

```
cov = cov.drop('target', axis=1)
```

Unfortunately, the features generated from the COVID-19 will not be 100% identical to the other coronavirus of our animals. We can observe that the number of features of the COVID-19 coronavirus is 409, which is a lot less when compared to the 1786 features of animal coronaviruses. Due to this mismatch, we are required to perform another 2 types of data transformation before the dataset is fully compatible with our prediction model.

[1] What are Features in animal coronaviruses that the COVID-19 coronavirus do not have?

This means discovering the features that are in the animals' nucleotide sequences, but they do NOT have a presence in the COVID-19 coronavirus. Once we found such features, we will need to populate them as "unknown" in our prediction dataset.

```
mc = X_train.columns.difference(cov.columns)
```

```
mc
```

We can see that the features "AAAM", "AAR", and "AAAY' are some of the features that we found in the coronaviruses, but they do not exist in the COVID-19 coronavirus.

It is a requirement to add these features so it matches with our training datasets. Therefore, we can populate the values as unknown. Since the features have to be in numeric form, we will use '-999' numeric value to represent the "unknown".

[2] What are the New Features of COVID-19 that never existed in other types of animal coronavirus?

Most importantly, we will also need to find features that exist in COVID-19 but do not exist in the training datasets. Once we found such features, we will need to remove those features to match them with our training datasets.

```
rf = cov.columns.difference(X_train.columns)
```

```
cov = cov.drop(rf, axis=1)
```

We can see that "AWGC", "CS", "CST", and "CSTG" are some of the new nucleotide regions of COVID-19 virus that have never existed before in the coronaviruses of the four animals.

After removing all these features from the predicted datasets, our prediction datasets finally match our training datasets.

Results:

We are now ready to predict the likely origin of the COVID-19 coronavirus! The next step is to predict the likely source of origin of the 34 samples of the COVID-19 coronavirus we collected. This can be done easily using the XGBoost predict API.

```
model.predict(cov)

array(['bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat',

'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat',

'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat',

'bat', 'bat', 'bat', 'bat', 'bat', 'bat', 'bat'], dtype=object)
```

Based on the predicted values, we can see that our model predicts all the 36 samples as Bat. Not even a single of the sample was predicted as other animals. To analyze this further, we will use the predict_proba API of the XGBoost to obtain the probability of each prediction.

```
print(model.classes_)

similarities = model.predict_proba(cov)

np.round(similarities, 3)

[‘bat’ ‘cattle’ ‘chicken’ ‘duck’]

array([[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],
```

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041],

[0.957, 0.001, 0.001, 0.041]], dtype=float32)

Based on the predicted probability, we can clearly see that the predicted probability of bat is extremely high with predicted values of 0.957. This implies that there is a 95.7% chance that the origin of the COVID-19 virus is from bats, while there is only a 4.1% probability that it comes from the duck. The probability that the virus comes from either cattle or chicken is only 0.1%. Therefore, we can strongly conclude that bats are the likely origin of the 2019 novel coronavirus.

Comparing our Results with BLAST

As mentioned at the beginning of the article, BLAST (basic local alignment search tool) is typically the standard method that allows scientists to compare differences of nucleotides or proteins sequences. This program is publicly available here: https://blast.ncbi.nlm.nih.gov/. From the website, you can upload your SARS-CoV-2 sequences as FASTA format, and the program will query your sequences against its database to find any similar sequences that match your query. Unlike our approaches, BLAST does not apply any predictive models on features, but rather it uses a heuristic search algorithm. One major benefit of our predictive approach is that it may discover more insight and it could find the hidden patterns that the standard heuristic search normally unable to uncover.

Discussion (Summary& Conclusion)

In this paper we discussed about similarity searching algorithms FASTA & BLAST, we shed light on the topic that how these two algotithms originated and how they they work and how they have become immensely useful for everyone involved in research studies in Bioinformatics. We used BLAST and Data science techniques for finding the origin of SARS-CoV-2 coronavirus and also compared the result of both the techniques**.** In the midst of the global COVID-19 public-health emergency, it is reasonable to wonder why the origins of the pandemic matter. Detailed understanding of how an animal virus jumped species boundaries to infect humans so productively will help in the prevention of future zoonotic events. For example, if SARS-CoV-2 pre-adapted in another animal species, then there is the risk of future re-emergence events. In contrast, if the adaptive process occurred in humans, then even if repeated zoonotic transfers occur, they are unlikely to take off without the same series of mutations. In addition, identifying the closest viral relatives of SARS-CoV-2 circulating in animals will greatly assist studies of viral function.

References

[1] R. M. Casey (2005). “BLAST Sequences Aid in Genomics and Proteomics”. Business Intelligence Network.

[2] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990). Basic alignment search tools. J. Mol. Biol. 215:403-410.

[3] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W Lipman DJ (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 25(17):3389-3402.

[4] Brenner SE, Chothia C, Hubbard T JP (1998). Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. Proc. Natl Acad. Sci. USA, 95:6073-6078.

[5] Lipman DJ, Pearson WR (1985). Rapid and sensitive protein similarity searches. Science, 227:1435-1441.

[6] Luscombe NM, Greenbaum D, Gerstein M (2001). What is bioinformatics? A proposed definition and overview of the field. Methods Inf. Med. 40(4):346-358.

[7] Zhou, P. et al. Nature https://doi.org/10.1038/s41586-020-2012-7 (2020).

[8] Alexander, D. J. & Brown, I. H. Rev. Sci. Tech. 28, 19–38 (2009).

[9] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic Local Alignment Search Tool. J Mol Biol 1990;215:403–410. [PubMed: 2231712]

[10] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997;25:3389–3402. [PubMed: 9254694]146917