POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's degree Thesis

Virus variants evolution via Machine Learning and epidemic Renormalisation Group

Supervisors

Prof. Francesco SANNINO

Prof. Elisa FICARRA

Prof. Francesco CONVENTI

Candidate

Adele DE HOFFER

Academic Year 2020-2021

Acknowledgements

Al mio relatore per avermi trasmesso la passione per la ricerca e l'amore per quello che sto facendo. Il destino a volte ti porta verso persone che non avresti mai immaginato, ma non avrei potuto chiedere niente di più.

Al mio correlatore, che ha avuto la malaugurata sorte di dovermi accompagnato per tutta questa analisi. Grazie per essere sempre riuscito a farmi sorridere anche quando non funzionava nulla e volevo solo lanciare il computer contro il muro (praticamente tutti i giorni).

Al mio gruppo di ricerca, per tutte le presentazioni in pessimo inglese che hanno dovuto subire e dovranno continuare ad ascoltare. Grazie per tutta la pazienza, la gentilezza e per insegnarmi ogni volta qualcosa di nuovo.

Alla mia stellina, per tutte le ore di web e per tutti gli scleri. Grazie per essere la mia sicurezza in questo momento di cambiamenti, grazie per avermi fatto capire che quando trovi qualcuno fatto per te la distanza non può essere un ostacolo.

Ai miei amici, per tutto l'affetto e la pazienza che servono per sopportarmi quotidianamente.

A Torino, che mi ha tolto tanto ma ha donato infinitamente di più.

Ai miei Salmons, con cui ho condiviso troppi CFU di elettronica ma troppe poche paste al pesto.

Ai miei Buddies, i miei deep learning experts preferiti, per essere stati i miei punti fissi in questi due anni. Grazie per non avermi mai fatto affrontare nulla da sola, senza di voi probabilmente in questo momento non sarei qui.
Alla mia collega preferita che sa quello che penso senza che le debba dire nulla,

qualcuno deve averci punito conferendoci gli stessi schemi mentali.. quella I di INF-ci ha fregato, ma non vorrei essere I con nessun altro.

Ai miei cuori, le mie anime gemelle da e per tutta la vita, la dimostrazione costante che l'amore esiste ed è una cosa meravigliosa.
Grazie perchè ovunque voi stiate, quella è casa mia.

Al mare, che ha saputo consolarmi quando le parole non sarebbero bastate.

Alla mia famiglia, quello strambo posto dove sono capitata e che amo da matti.
Grazie per sostenermi sempre, anche quando non mi capite.
Ma soprattutto, grazie per amarmi anche quando sento di non meritarlo.

A mio fratello, il detentore del digi-medaglione del coraggio, la mia persona preferita e quella a cui tengo di più al mondo.

Ho avuto la fortuna sfacciata di essere circondata da persone incredibili, che per qualche strano motivo hanno deciso di credere in me ancor prima che lo facessi io. Non capisco mai bene cosa vediate in me, ma mi impegno tutti i giorni affinché tutta questa fiducia non venga sprecata.

E infine a me stessa, a tutti i piantini, a tutte le crisi di identità e a tutte le volte che pensavo di non farcela, di mollare tutto e scappare lontano.
A tutte le volte che mi sono persa, e poi ritrovata.

A volte mi sento di star vagando su una strada non battuta e piena di buche che, a dirla tutta, non ho ancora ben capito dove debba portare. Ci sono salite che mi sembrano interminabili, discese troppo ripide e appena penso di aver intrapreso il percorso giusto compare qualche deviazione inaspettata che finisce per portarmi da tutt'altra parte. Non capisco mai in che direzione stia andando e i pochi cartelli che compaiono sembra siano messi lì solo per confondermi di più.

Tutto varia, troppo velocemente per riuscire ad afferrarlo, ma fortunatamente basta che mi guardi un attimo attorno per vedere che in mezzo a tutto questo caos c'è qualcosa di stabile.

Se non sono mai rimasta pietrificata a piangere sul bordo della strada, se non sono mai rimasta a terra dopo una brutta caduta è perché appena alzo gli occhi dal terreno incontro subito i vostri.

Perché si, è una strada difficile e tortuosa ma voi avete sempre fatto in modo che non la dovessi mai percorrere da sola, accompagnandomi in ogni singolo passo.

Grazie per essere i migliori compagni di viaggio che potessi desiderare, con voi anche una fifona come me riesce a superare le sue paure.

Quindi congratuliamoci a vicenda perché se siamo arrivati fin qui...
è stato un lavoro di squadra!

La vostra Dende

Summary

In an unprecedented way, humans have been able to collect a vast amount of high-quality data for the Covid-19 pandemic. These include its sequenced genome and proteome, together with relevant metadata such as geolocation and time evolution. Henceforth, advanced data analysis techniques like machine learning can be exploited to maximally rip useful information stemming from the huge amount of data collected in search of precious hidden patterns.

Historically, epidemiological models are built on SIR: a compartmental model that dives the population into three subgroups (susceptible, infected and recovered) and then studies the evolution of the system through first order derivative equations that explain the interaction among these groups. However this kind of models are not accurate to describe real word situations in which the variables under analysis are more than one. So, to overcome this limitation, it has been introduced an epidemic Renormalisation Group approach that aims to define the evolution of the system by applying concepts of theoretical physics. In particular, this approach exploits temporal symmetries and scale in variance to describe the temporal behaviour at the global level by the auxiliary of a β function. Moreover, this model highlights and demonstrates that each pandemic wave is guided by a new emerging variant.

The envisioned project aims to yield a profound understanding of the genesis, dynamics, and evolution of the virus variants. The ultimate goal is to develop a mathematical model of virus evolution, combining our machine learning techniques with underlying time-dilation symmetries. The latter constitute the backbone of the recently introduced epidemiological Renormalisation Group framework.

Looking not only at the protein itself but at its evolution through time, we can build, from scratch, a novel evolutionary model. This will allow us to answer fundamental questions about the nature and dynamics of the virus, ranging from how fast it mutates to which mutation have a chance of becoming dominant, or even whether it is possible to identify and predict which mutations have a chance of

becoming the potentially dangerous ones. Specifically, it is relevant to concentrate the data analysis on the spike protein of the virus since it plays a key role in the virus interaction with the human body, as a matter of fact the virus infects the body by binding the receptor ACE2 on the surface on the lung cells. In this sense an early detection of pattern of potentially dangerous mutations could enhance the health system capability to prompt react.

Concretely, I developed an advanced state-of-the-art machine learning technique for such analyses in the context of viral genesis and evolution.

I started the data analysis using the time ordered amino acids sequences of the SARS-CoV-2 spike protein (that can be seen as a list of around 1200 characters) retrieved from the open-source website GISAID.

The first step of the procedure is devoted to the reconstruction of the time series of the spike proteins sequences: to do so I divided the whole dataset into monthly time bins and for each time step I performed a cluster analysis to group together similar sequences. The procedure proposed is based on the hierarchical clustering principle, built via the Levensthein distance, the latter is defined as the minimum number of characters needed to transform the string corresponding to one variant into another one. In fact, a sequence of amino acids can be considered as a string of characters constituted by amino acids. The idea of hierarchical clustering is to build a tree of samples by merging together first the nearest sequences and then proceeding with the next ones, after that I needed to tune the relevant threshold parameters namely the value used to cut the tree and the minimum number of elements needed to define a cluster.

Indeed the clustering procedure split the dataset into time steps and each time step has a given number of clusters. The main goal of this procedure has been the development of a criteria to connect clusters in consecutive months in order to reconstruct the evolutionary paths of similar sequences. In this sense, I defined a method to link together clusters in consecutive months via strong or weak link based on the distance among them. Two or more consecutive clusters linked together are called chains. Moreover I developed a dedicated mechanism to determine if a chain can be derived from another existing chain using the first cluster in the chain and looking for similar clusters in the previous month in terms of parenthood.

This way I succeeded to create chains of clusters that well represent the evolution of different variants of the spike protein, with results coherent with the history of the pandemic.

The robustness of this approach relies on the possibility to build time evolution of candidate variants without any prior bias moreover it is also possible to recover time information of a candidate variant, and for each candidate variant it is also possible to define a parent variant to better track the evolutionary path. This way the proposed method allow to identify some variants of concern and interest together with their evolutionary path and it represent a first attempt towards the development of more complex strategies to study the evolution of variants in the current pandemic.

In fact, at the end we are able to successfully identify the evolutionary pathway of some variants of concern (like the alpha and the delta VoC) and interest, together with other variants that were not identify with the standard methods. At the end, we are able to decompose each pandemic wave into sub waves,and see the contribution of each variant to the total number of infected. Moreover the approach achieves very good results compared to variants classification from WHO and GISAID and presents a lighter tool compared to the standard ones that use all the genome sequence of RNA.

The machine learning analysis allows us to naturally integrate the temporal evolution of virus variants and their genesis into the eRG framework discussed at the beginning. This leads to a coherent picture of how temporal symmetries are key to understand not only the overall epidemiological understanding of a pandemic but also its atomic version in terms of the virus variants. In other words we have now an epidemiological theory of variants based on fundamental physics principles.

Specifically, this work is the first part of a more ambitious project that aims at creating a deep learning algorithm to predict future mutations: the first part of this scenario was in fact the construction of the variants' pathway.

The work is articulated in the following way:

1. In the first chapter I introduce the epidemic framework starting from the SIR model and how it can be modified and used to construct the epidemic Renormalization Group, then there is a short description of the spike protein of SarS-CoV-2.

2. In the second chapter a brief overview about deep learning, the machine learning algorithms needed for the analysis together with the techniques for data reductions have been presented.

3. The third chapter is the core of the epidemic analysis on the spike protein. It can be schematized in a first part in which the clusterization is performed and a second part in which these clusters are linked together and studied.

4. The last chapter highlights the conclusion of the project and proposes some possible future works, in particular the last section represents the backbone for the implementing of a deep learning algorithm (LSTM) to predict further mutations.

Table of Contents

Chapter 1

Introduction

1.1 Covid-19 and SARS-CoV-2

Covid-19 is the disease caused by severe acute respiratory syndrome coronavirus 2 also known as SARS-CoV-2. The first case of infection was registered in Wuhan (China) in December 2019, but the infection spread rapidly and around February it has already became a world problem.

SARS-CoV-2 arises as a single strain RNA virus belonging to the family of coronaviruses, which contains viruses from ordinal flu, MERS (Middle East Respiratory Syndrome) and SARS (Severe acute respiratory syndrome). Usually the reservoir of these viruses are animals, in fact the most likely hypothesis is that the human coronavirus is derived by a spill-over event from bat coronavirus.

The main symptoms are fever, cough, difficulty to breath and the lost of taste and smell. The gravity of the infection depends on the individual: it can be asymptomatic in the youngest cases, although in the worst cases it can even lead to pneumonia and hospital recovery.

The virus passes from one host to another by respiratory droplets, and then enter into the lung cells by binding the receptor ACE-2 by one of its membrane protein called Spike. Then the virus uses the host cell to replicate its genome, its structure and assemble them to create new viruses. At the end, it forces the cell to die, the copies of the virus are released and then they are they free to continue the infection using in this chain mechanism. The incubation time for this kind of virus is until 14 days, in this sense it it very luckily that the host will have the possibilities to spread the virus or anyhow transport it for long distances.

In this scenario the aim of this work is to investigate the variants responsible for the ongoing pandemic and define some method to track their evolutionary pathway. In this sense, we are trying to answer fundamental questions about the genesis and the role of this kind of mutation in Covid-19 pandemic.

To do so, we will focus our analysis on the the spike protein of SARS-CoV-2, in fact, as we will see, it represents a key factor for the virus interaction with the human body. Therefor, investigating its evolution may be a good starting point to better clarify the dynamics that lead to the current pandemic.

1.2 Virus Structure: the Spike protein

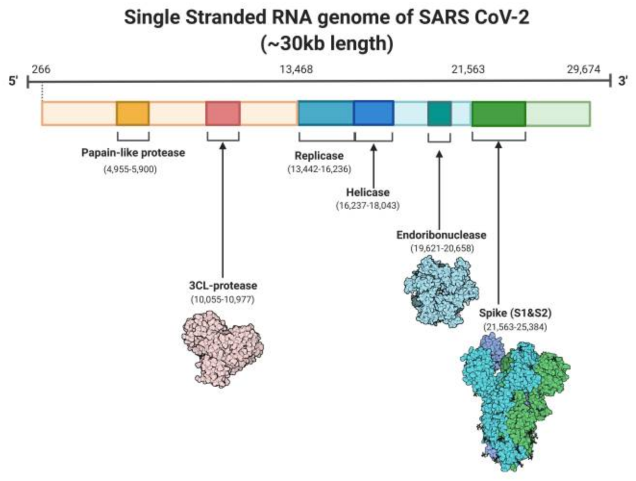


Figure 1.1: Overview of the regions of RNA [1]

SARS-CoV-2 stands for virus of severe acute respiratory syndrome–related coronavirus. It is single strand RNA virus, it composed by a single positive strain of RNA containing around 30’000 nucleobases that codifies for 7 main proteins. In particular, 4 of them (S, E, M and N) are structural proteins: the first three create the envelope of the virus, besides the last one holds the strain of RNA.

The structure of the virus and its proteins can be appreciated in Figure 1.2, where we can see that proteins S,E,M are on the surface of the cell and the N protein is holding together the RNA strand. This analysis is focused on the

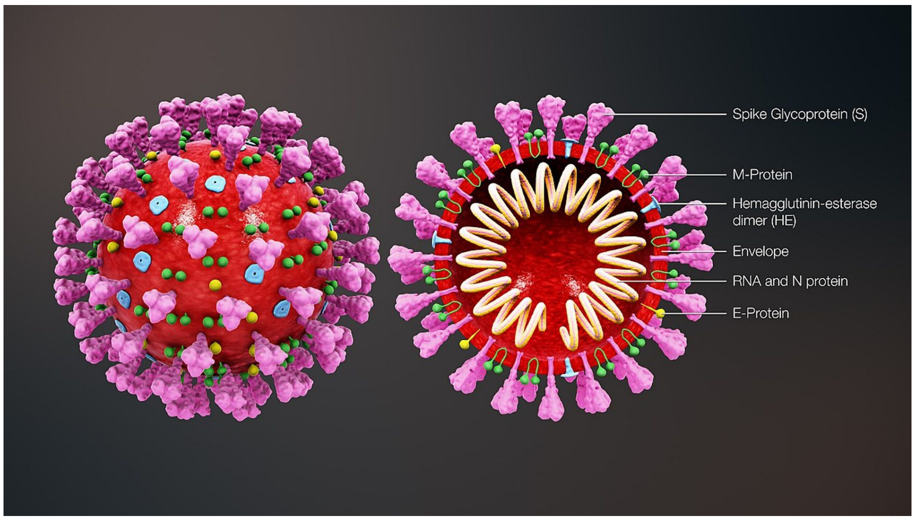


Figure 1.2: Sars-CoV-2 Overview [2]

S protein (Spike), which is the key factor for the virus interaction with the lung cells.

The protein is composed by a chain long 1273 amino acids, that defines its structure and properties, besides to function correctly it should be assembled into a trimer that has the form of a crown (as also shown in Figure 1.2), hence the name 'Coronavirus'.

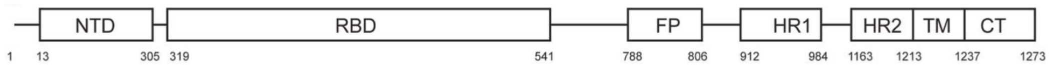


Figure 1.3: Spike protein domains [3]

In figure 1.3 we can see the amino acid coordinates for different domains of the Spike protein [4].
In particular:

- **NTD:** N-terminal domain, part of the trimer interface
- **RBD:** Receptor-Binding Domain, the responsible for the binding with the ACE2 receptor
- **FP:** Fusion Peptide
- **HR1:** Heptapeptide Repeat sequence 1
- **HR2:** Heptapeptide Repeat sequence 2

- **TM:** Transmembrane Domain
- **CT:** Cytoplasm Domain

Moreover, the structure of the spike protein is divided into 2 main sub-units called S1 (from 0 to 541) and S2 (from 542 to 1273), which are cleaved to permit the protein to function properly [5] as shown in Figure 1.4.

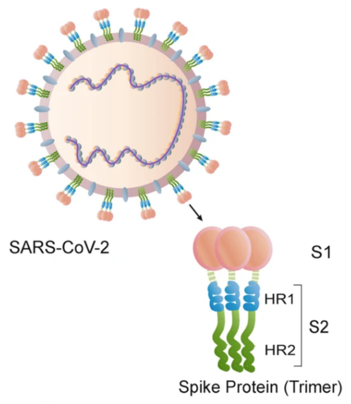


Figure 1.4: Spike sub-units [3]

S1 is responsible for binding the ACE2 (Angiotensin Converting Enzyme 2) receptor on the lung cells, instead the S2 part is the one that allows the virus entrance inside the cell membrane as shown in Figure 1.5.

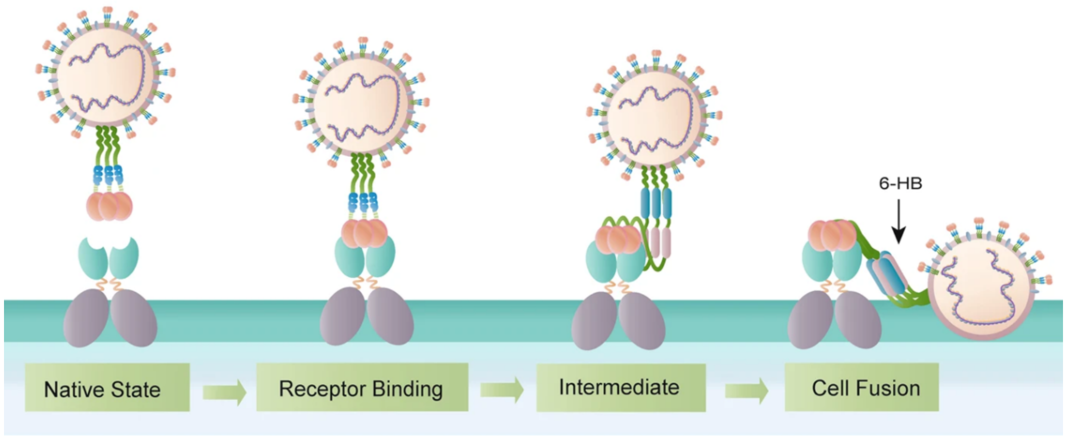


Figure 1.5: Spike entrance mechanism [3]

SARS-CoV-2 is a virus with a low mutation rate [6] , it means that its nucleotides

and aminoacid sequence tends to preserve itself.
For this analysis it is useful to clarify that we define:

- **mutation** as a single aminoacid substitution [7]
- **variant** as a unique strain (sequence) of spike protein that contains mutations

1.3 Epidemiology

Epidemics are always more present now days, also due to the increasing of the population and the ecological footprint humans are leaving behind.
Starting from the Spanish Flu [8] and ending with the ongoing pandemic of Covid-19.
So it is important more than ever having strong mathematical tools that are able to describe and predict the evolution of these kind of infections.

Following the approach of [9] the epidemiological theories are here divided into:

- **Stochastic:** Based of the fact that all the microscopic interactions between individuals are ruled by a probabilistic nature. Like graph theory over SIR, where the final state of the system is the result of a chain of stages of probabilistic nature, so changing one of these aspects may lead to a very different results. In this case, to obtain a confident prediction average operation must be performed.
- **Deterministic:** The behaviour of the global system is a process that can be predicted by differential equations

Here a brief summary on SIR, together with some adjustment needed to adapt it to a real world situation. And how we can start from this model to derive a new mathematical approach named eRG that exploits temporal symmetries and scale invariance to describe the system at a global level.

1.3.1 SIR

SIR [10] is a general model for the diffusion of epidemic, this is a compartmental model that divides the population into 3 non overlapping subgroups:

- **S: susceptible**
  People that can contract the disease
- **I: infected**
  People who have the disease, they can both infect susceptible people and recover

- **R: removed**
  People that recovered for the disease and can't contract it anymore or are removed from the analysis for other motives (isolation, death etc)

Each individual can be seen as an agent that at every time t is associated to a state among **S,I,R**.
The state of each agent at time $t+1$ depends only on its spontaneous mutation and on the states of its neighbors at time t.

In fact, an agent in S may become I by interacting with agents in state I with probability β. Besides, an agent in state I can both infect agents in S and spontaneously recovery with probability γ. Instead, an agent in state R cannot change its state or transmits anything.

Where β is the infection rate, it means how probably it is for an infected person to transmit the disease to a susceptible one.
In other term, the probability of an agent on state S to become I at time t after being exposed to an agent in state I at time $t-1$.
On the other hand γ is the recovery rate: the probability that an infected individual heals (or dies etc) and be removed, so to have a transition from I to R. Notice that this probability depends only on the state of the agent and not on its neighbors.

So, the following transitions are permitted:

$$S \to I \to R$$

Considering N total individual as constant (the number of people does not change during the analysis), we can normalize the quantities S, I, R write for each time t:

$$\begin{aligned} & S(t) + I(t) + R(t) = 1 \quad \forall t \in R_+ \\ with \\ & \tilde{S}(t) = NS(t) \\ & \tilde{I}(t) = NI(t) \\ & \tilde{R}(t) = NR(t) \end{aligned} \tag{1.1}$$

We need N to be large enough to consider $S(t), I(t) and R(t)$ as continuous in time. Looking at the dynamics of the system, we can write the transition from time t to time $t+1$ as function of the global number of individuals in each state:

$$\begin{aligned}
&\frac{dS}{dt}(t) = -\beta I(t)S(t) \\
&\frac{dI}{dt}(t) = \beta I(t)S(t) - \gamma I(t) \\
&\frac{dR}{dt}(t) = \gamma I(t)
\end{aligned} \tag{1.2}$$

Looking at figure 1.6, we can see a summary of what happens for a generic SIR model. At the beginning (without losing generality we can set $t = 0$), all the population is in state S, but after that some individuals become infected the spreading of the disease starts.
The number of I reaches a peak and then it decreases again. At the end, in this graph all the population is in state R.

For our further analysis is very important to take into account the fact that the cumulative number of I can be model as a sigmoid function.

$$I_c(t) = NI_0 + \int_0^t dt'\beta NI(t')S(t') \tag{1.3}$$

1.3.2 SIIR

The basic SIR model answers many questions but it fails to describe a more real situation in which the variants under analysis are more than one. So, we need to adjust this model to take into account different variants, in fact as a virus mutates it can be considered as a new virus that starts competing with all the other ones for the surviving.

To describe this situation in [12], it is described a new type of compartmental model in which we have 2 types of infected individuals:

- I_1**:** people infected with variant 1
- I_2**:** people infected with variant 2

We assume that once an individual recovers from the disease i, it cannot be infected with the other one. In this sense, every individual in state I_i becomes R without taking into account if they are infected with variant 1 or 2, this means that we are excluding re-infections. This situation is well explained in Figure 1.7,

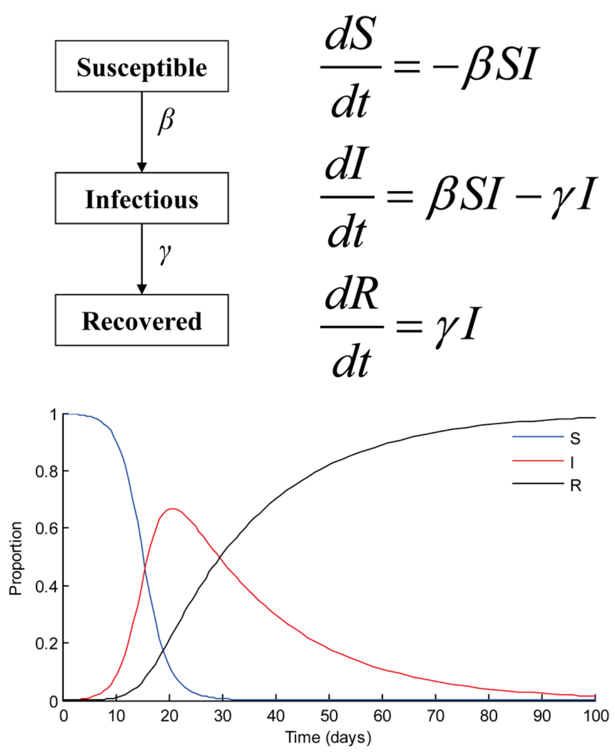


Figure 1.6: Summary of SIR model [11]

considering the same assumption we had in Section 1.3.1.

As said before, this two variants compete for the spreading in the population, now we can look at the temporal evolution of these 2 variants.

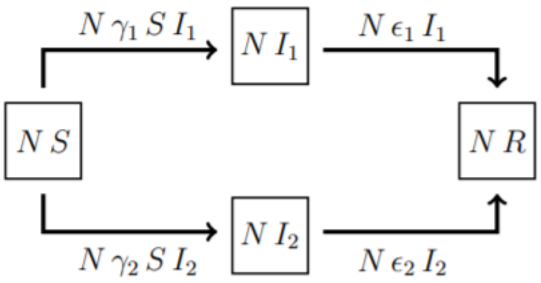


Figure 1.7: SIIR [13]

Naturally, each variant has its β_i and its γ_i, so we can rewrite equation ... as:

$$\begin{aligned}
&S(t) + I_1(t) + I_2(t) + R(t) = 1 \quad \forall t \in R_+ \\
&\frac{dS}{dt}(t) = -(\beta_1 I_1(t) + \beta_2 I_2(t))S(t) \\
&\frac{dI_1}{dt}(t) = \beta_1 I_1 S - \gamma_1 I_1 \\
&\frac{dI_2}{dt}(t) = \beta_2 I_2 S - \gamma_2 I_2 \\
&\frac{dR}{dt}(t) = \gamma_1 I_1 + \gamma_2 I_2
\end{aligned} \tag{1.4}$$

Moreover, the the model we want to write is based on the assumption that variant 2 is a mutation of variant 1. So, at the beginning there are only individual infected with variant 1, then at a random time t_0 some individuals will be infected with variant 2.
We want to see how the dynamics of the system evolves under these assumptions.

Looking at a standard SIR model we can introduce the reproduction number R_0, as the ratio between the infection rate and the recovery rate. We can visualize R_0 as the expected number new infections that will be produced by an infected individuals before its recovery:

$$R_{0,1} = \sigma_1 = \frac{\beta_1}{\gamma_1} \qquad R_{0,2} = \sigma_2 = \frac{\beta_2}{\gamma_2}$$

The authors show that the behaviour of the first variant is changed by the second variant only if σ_2 is larger than σ_1. So, if the two variants have similar reproduction number we can consider the cumulative number of infected individuals of variant i indipendent from the other one.

1.3.3 epidemic Renormalization Group Theory (eRG)

This overarching idea of the approach derives from theoretical physics. The logic is to describe complex systems in terms of fewer degrees of freedom making use of the powerful renormalization group approach augmented with identifying important symmetries of the problem. In epidemiology time-scale (quasi) invariance is the symmetry to implement to re-organize the description of the epidemic diffusion process.

We have scale invariance when the diffusion is stable, this means that the total number of cumulative infected individuals becomes a constant.

Concentrating, for example, on the overall duration of a pandemic wave one can integrate out the short-time degrees of freedom and concentrate on space-averaged quantities such as the total number of infected individuals, de-facto becoming the degrees of freedom needed to effectively describe the temporal evolution of the process [14]. Interpreting this time-dependent quantity as an operator in field theory one can construct a single differential equation that is called flow equation. In this way one can capture in an highly efficient manner the description of the time evolution of the system, and reducing the computational complexity.

The behaviour of the system is described in terms of fixed points [15], which are the points at which the differential equation is 0.

The system is flowing from one fixed point to another one, which are the 'phases' of our system. In this term, taking into account only one variant we are observing to a phase cheange from 0 infected to A infected. The approach is known as the epidemic Renormalisation Theory (eRG) [12] that efficiently captures the temporal epidemiological dynamics [16].

Borrowing from theoretical physics the first-order differential equation defines a β function that rules the dynamics of the system at a global level. In particle physics the beta function encodes the dynamics of a given theory with the number of infected replaced by the coupling strength of the model. The latter defines a map between the coupling and the scale of the theory which is bijective, monotonic, differentiable.

In fact, we select $\alpha = f(I_c)$ with I_c cumulative number of infected individuals Here we can define the β function as:

$$-\beta(I_c) = \frac{d\alpha}{dt}(t) = \lambda\alpha(1 - \frac{\alpha}{A}) \quad with \quad \alpha = f(I_c) \tag{1.5}$$

This equation has 2 fixed point for $\alpha(t_0) = 0$ and $\alpha(t_0) = A$, this means that in these cases the system will be remaining stable.

The solution of (1.5) is a sigmoid function described as :

$$\alpha(t) = \frac{A}{1 + Be^{-\lambda t}} \quad \alpha : \mathbb{R} \to [0, A] \tag{1.6}$$

It is also shown in Figure 1.8, as the beginning the number of infected I_c is around 0 and it increases dramatically until it reaches the constant value of 20000 that saturated the graph. In this sense, we can say that we are assisting to a phase transition in function of t, which in this case is discretized. And as expected:

$$\lim_{t\to-\infty} \alpha(t) = 0 \quad and \quad \lim_{t\to\infty} \alpha(t) = A \tag{1.7}$$

In particular,

- λ takes into account how fast the infection is spreading
- **A** is the value of the asymptotic value of the total number of infected
- **B** is a shifting constant that determines the beginning of the infections

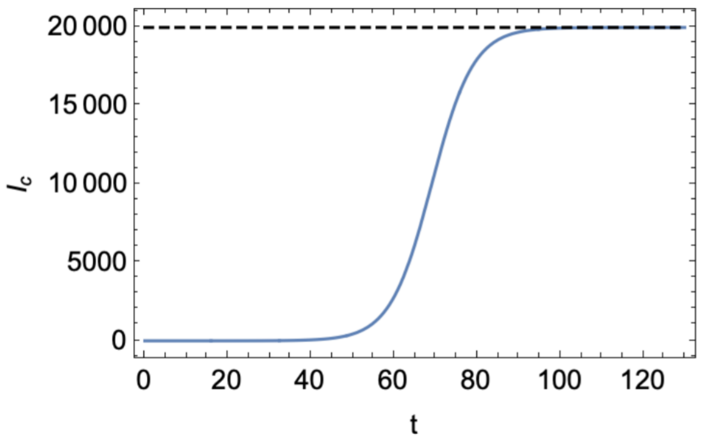


Figure 1.8: Sigmoid [9]

The system is flowing from one fixed point in which we have 0 infections toward another fixed point in which the cumulative number of infected individuals becomes a constant.

As we have seen, we are able to model the spreading of the disease using just one differential equation!

1.3.4 Multi wave

The latter section it is true taking into account just one variant, but we want to study the behavior of the system when it is subjected to more than one disease.

Here, we are considering the case in which two variants 1 and 2 are present.

In this case we need to define an α for each variant. So we have $\alpha_i = f_i(I_{c,i})$ with $I_{c,i}$ cumulative number of infected individuals with variant i.
And we can define the β function as:

$$-\beta(I_{c,i}) = \frac{d\alpha_i}{dt} \quad \forall i = 1{,}2 \tag{1.8}$$

So the latter becomes:

$$-\beta(I_i) = \frac{d\alpha_i}{dt} = \nabla_i \Phi(I_{c,j}) \quad with \quad \Phi(I_{c,j}) = \sum_{j=1}^{2} I_{c,j}^2 \frac{\lambda_i}{2}(1 - \frac{2I_{c,j}}{3NA_j}) \tag{1.9}$$

This under the assumption that the two σ are not too different from each other, in this way we can see that the cumulative number of total infected of a variant i is independent from the other one.
This assumption comes from the real word data, in fact as said before we can model the number of total infected individuals as a logistic function:

$$I_{c,i}(t) = \frac{NA_i}{1 + e^{-\lambda_i(t-\kappa_i)}} \quad \forall i = 1{,}2 \tag{1.10}$$

The latter highlights that each pandemic wave can be model by an independent eRG.

Always considering the case in which only 2 variants are present we can look at the dynamics of the system exploiting the RG flow, in particular we can see that we are dealing with 4 fixed points (the dynamic of the system stops when it reaches one of these points) as shown in the plane

$$P = \{(I_{c,1}, I_{c,2}) \in [0, N] \times [0, N] | I_{c,1} + I_{c,2} \leq N\} \tag{1.11}$$

depicted in Figure 1.9

- $(0{,}0) = P_0$
- $(NA_1{,}0) = P_1$
- $(0, NA_2) = P_2$
- $(N_A 1, NA_2) = P_3$

The trajectory that goes to one fixed point to another is guided by the flow depicted in the graph.
As seen, at the beginning we have the situation P_0 in which we have no infection at all. This fixed point is repulsive: once the system exits from this equilibrium it will never come back.
After leaving this point, the dynamics is ruled by the arrows and the depicted in the graph and the system is forced to follow them.

The other two fixed points are the situation in which the individuals are infected only by a variant, these points are attractive in one direction and repulsive in the

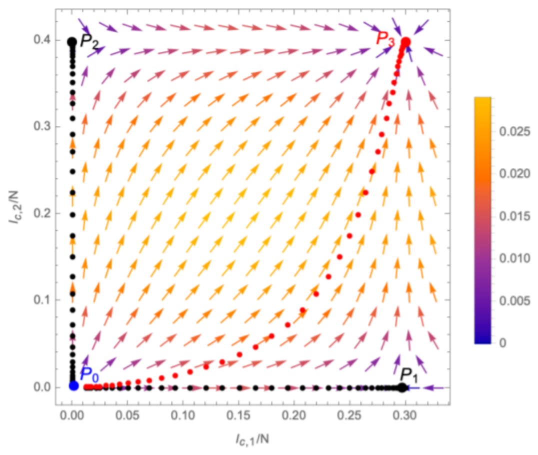


Figure 1.9: eRG flow [13]

other one, this means that the system can reach and leave these points.
Another fixed point is (NA_1, NA_2) that represents the situation in which the two variants coexist and it is attractive for all the directions.

We define as critical surface for the fixed point P_i all the points on the plane that lead to a this fixed point. The arrows reflect the dynamics of the system for every point in the graph. In particular, we can see that we can reach the fixed points P_1 and P_2 only by the lines that connect them with the initial fixed point P_0.

For simplicity, the authors highlight a possible path in red, that leaves the point P_0 and then it starts to goes toward P_3.

Furthermore we can define:

- **Relevant Operator:** direction that drives the dynamics away from the fixed point (this can happen for the system that from P_0 is going toward P_1 or P_2)
- **Irrelevant Operator:** direction that does not drive the dynamics away (in this case all the direction that arrives at P_3)

Now that we have all the elements needed we cam see how to model the spreading 2 variants in the multi wave scenario [13]!

Considering the case in which the new variant 2 emerges from the variant 1, the situation is the following: from the initial fixed point (0,0) it appears a first deformation that introduces the variant 1, after a pretty long stage a second deformation occurs and introduces variant 2. So as shown in figure 1.10 at the beginning the dynamic is flowing toward the fixed point P_1, then a small perturbation introduces a relevant operator along the perpendicular direction and forces the system to go toward P_3.

In particular, the authors inspect the case in which the second deformation appears after that I_1 reaches the maximum of the cumulative number of infected individuals. This situation is called *crossover flow* and the dynamics is shown in figure 1.10 . Here, the flow arrives in the proximity of the fixed point P_1 and

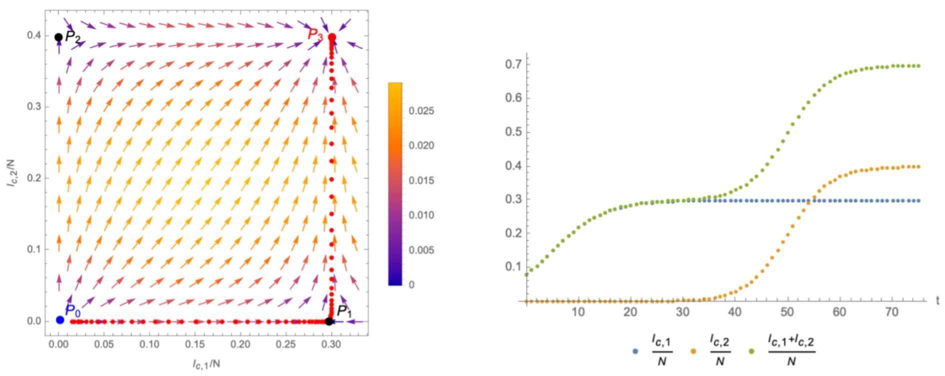


Figure 1.10: Crossover Dynamic [13]

then the system enters into a quasi-linear growth phase where the number of new infections is small for both the variants, and the cumulative total number of infected grows linearly.

After a while, the number of cumulative infected for the variant 2 begins to grow exponentially and here we enter into the *crossover phase* and we can again model the phenomena with eRG dynamics.

The linear growth depends on the fact that the system is in the proximity of a complex fixed point but it cannot reach it, since the cumulative number of infected $I_{c,i}$ needs to be a real number.

Results are validated on real word data for the ongoing COVID-19 pandemic. At the end, the authors demonstrated that each pandemic wave is guided by a new variant. And each wave can be modelled by an independent eRG

Here, I reported few examples:

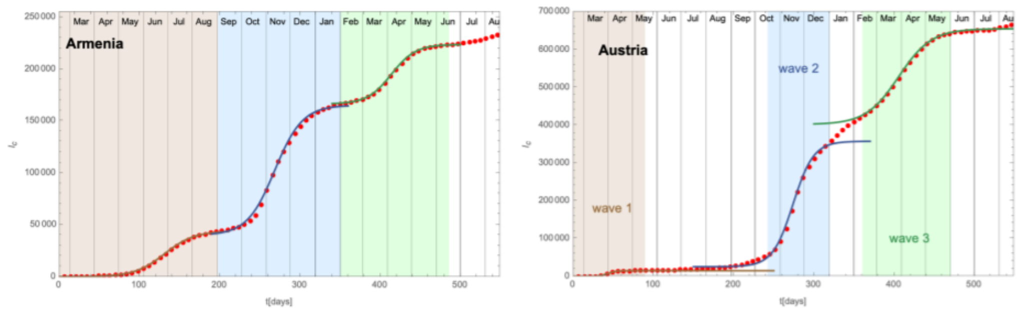


Figure 1.11: Multi wave pattern [9]

Chapter 2

Methodology

2.1 Distances Measures

The first thing to do it is defying the distance between two samples, in fact it will be a key factor for all the analysis.

2.1.1 Levensthein Distance

The Levensthein distance [17] is a metric that is usually exploits in linguistic context, it measures **the distance between two string as the minimum number of single-character edits needed to transform one string into the other one.** Type of edits:

- Insertion
- Deletions
- Substitution

For example:

HOUSE
MOUSE

The two string are the same apart from one character substitution, so the Levenshein measure between these two words is 1.

More in general, considering two string a and b:

$$lev(a,b) = \begin{cases} |a| \quad if \quad |b| = 0 \\ |b| \quad if \quad |a| = 0 \\ lev(tail(a), tail(b)) \quad if \quad a[0] = b[0] \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} \quad otherwise \end{cases} \tag{2.1}$$

This function is already implemented in Polyleven.

2.1.2 Pairwise alignment

It is a more sophisticate way to compare two strings. Here in particular, Bio python pairwise alignment [18] is used: the function takes as input 2 strings and returns their best alignment. The concept is the same of the Levensthein distance but now each operation we can use to edit the two strings is associated to a score:

- **Match:** same letters in position i
- **Mismatch:** different letters in position i
- **Gap insertion:** inserting a gap to reestablish the alignment
- **Gap elongation:** elongate the gap to reestablish the alignment

The aim of the function is to find the alignment that maximizes the global score, which is given by the sum of the scores associated to each edit performed. Example:

$$("ACCGT", "ACG", 2, -1, -.5, -.1)$$

Which gives:

ACCGT

A-CG-

$$Score = Match \cdot 2 - Mismatch \cdot 1 - Gap\_insertion \cdot 0.5 - Gap\_elongation \cdot 0.1 = 2 \cdot 2 - 2 \cdot 0.5 = 3$$

In fact, we can consider the Levensthein distance as a special case of pairwise alignment in which the score associated to each operation is 1.

2.2 Clustering

Clustering is an unsupervised machine learning technique that aims at dividing samples into subgroups (clusters) based on their distances, in this way elements that are similar are also grouped together.

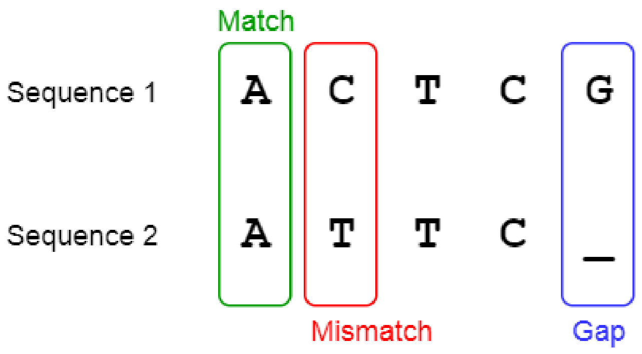


Figure 2.1: Pairwise Alignment Example [19]

2.2.1 Hierarchical Clustering

Hierarchical clustering [20] is divided into:

- Bottom up - Agglomerate
- Top down - Divisive

Agglomerative hierarchical clustering [21] starts to merge elements that are more similar and then incrementally build the clusters.
The algorithm works in this way:

1. At the beginning each element represents a cluster
2. Merge together the 2 nearest clusters
3. Create a new centroid for this element
4. Repeat from 2 until there is only one cluster

This procedure can be mapped into a dendrogram, in which each leaf represents an element and the values on the y axis the distance among elements. We can see in figure 2.2 that nearest elements are connected at lower values of distance and so on, at the end we have only one big cluster that contains all the samples.
The measure on the y axis is the distance between 2 clusters A and B is a parameter of the algorithm and can be chosen among the following:

- **Single Linkage**: we consider the minimum distance among all the elements in the two clusters

$$dist(A, B) = \min_{x \in A, y \in b} d(x, y) \tag{2.2}$$

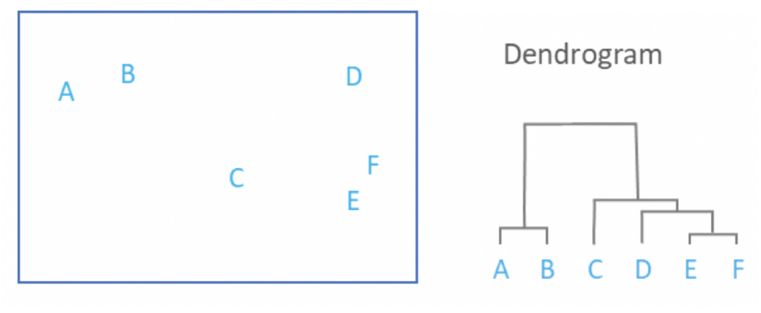


Figure 2.2: Hierarchical Clustering [22]

- **Complete Linkage**: we consider the maximum distance among all the elements in the two clusters

$$dist(A, B) = \max_{x \in A, y \in b} d(x, y) \tag{2.3}$$

- **Unweighted Average Linkage Clustering**: the average distance among all the elements in the two clusters

$$dist(A, B) = \frac{1}{|A||B|} \sum_{x \in A\,,\, y \in B} d(x, y) \tag{2.4}$$

- **Ward Distance**: this is the measure of the average squared distance of points in the cluster to its centroid. Hence, the effective distance between two branches is defined by the increase in the above measure in the merged cluster with respect to the two separate ones

$$dist(A, B) = \frac{|A||B|}{|A| + |B|}[\sum_{x \in A, y \in B} \frac{d(x, y)^2}{|A||B|} - \sum_{x, x' \in A} \frac{d(x, x')^2}{2|A|^2} - \sum_{y, y' \in B} \frac{d(y, y')^2}{2|B|^2}] \tag{2.5}$$

We can decide at which distance 'cut' the tree. This is basically to draw an horizontal line on the dendrogram at the value of the distance we want to consider, and then look at the different clusters we have.
Here, for example as seen in figure 2.3 we cut the tree at height equal to 5. So, the elements below that threshold are considered as one cluster. Hence, here we have 4 of them.

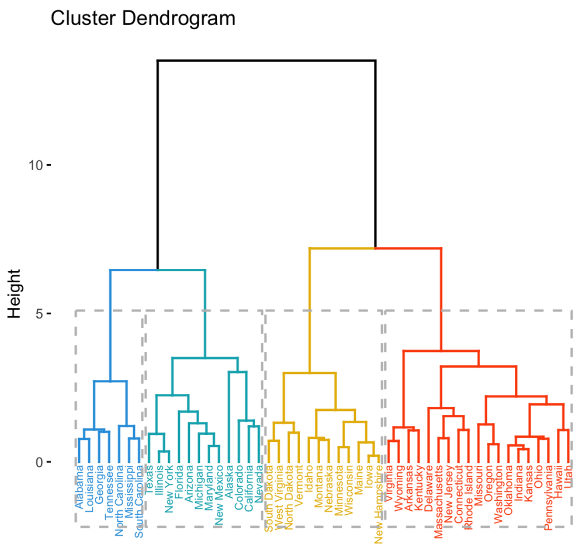


Figure 2.3: Dendrogram [23]

2.3 Feature Reduction

2.3.1 PCA

PCA [24] [21] [25] is a dimensionality reduction tool that aims to reduce the dimensionality of a matrix preserving as much information as possible.

The procedure maps a matrix x m of dimension m into matrix Wx of dimension k with $k < m$, so the goal is to find a space $k < m$ that maintains most of the data variability. Concretely it is a linear dimensionality reduction approach that performs

$$x \rightarrow Wx \quad where \quad W \in R^{mxk} \quad and \quad k < m$$

Basically we can write each instance of the original matrix in another way

$$x_i = a_1\psi_1 + a_2\psi_2 + .. + a_k\psi_k + a_{k+1}\psi_{k+1} + ... + a_m\psi_m$$

The new variables ψ_i are a linear combination of the original features and form an orthonormal basis which define a new coordinate system.

The ψ_i are called loading vectors and are the (ordered) component along which the data vary the most,a_i are the weight associated to each leading direction and represent the projection of the original instance on that vector.

We can decide approximate this mapping by cutting the elements to its k-th value. In this way we are reducing the dimensionaly but since the loading vectors were ordered considering their contribution to the variance of all the dataset, we are preserving the k most significant dimensions.
Basically, it is just a change of basis where the vectors are ordered and then an approximation.

To find the right ψ_i we can use the linear recovery vector. Defying: $y = Wx$, the linear recovery vector is $\tilde{x} = Uy = UWx$. This represents how much we can retrieve of the original sample by its approximation.
At the end it is just an optimization problem, in which we want to minimize the difference between the true value and its retrieve.

$$\arg \min_{W \in R^{n,d}, U \in R^{d,n}} \sum_{i=1}^{m} ||x_i - UWx_i||^2 \tag{2.6}$$

The solution is to set U to be the leading eigenvectors of $A = \sum x \cdot x$ Also, we can decompose the total variance of the data, ie we can see the percentage of the information that is expressed from each new component. In this case the first component ψ_1 retrieve the most part of the information, the component ψ_2 the second part and so on.

2.3.2 t-SNE

The t-distributed stochastic neighbor embedding (t-SNE) [26] is a non linear dimensional reduction algorithm that aims to reduce the dimension of a matrix preserving the local neighborhood. For this reason, it is usually exploited as visualization tool.

The aim is to preserving the clustering in the high dimensional space also in the lower dimensional space, in this sense we are preserving the local structure instead of the global one. Elements that are near in the higher dimensional space should be near also in the lower dimensional space.
The first step is to encode the samples x_i (of the higher dimensional space) into a distribution of probability.

Given 2 instance x_i and x_j, the probability of similarity between them is:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \quad with \quad p_{j|i} = \frac{exp(\frac{-||x_j - x_i||^2}{2\sigma_i^2})}{\sum_{i \neq k} exp(\frac{-||x_k - x_i||^2}{2\sigma_i^2})} \tag{2.7}$$

Where $||\cdot||$ represents the Euclidean Distance, and σ is a parameters of the algorithm.

Then the algorithm looks for points y_i in the lower dimensional space that preserve these similarities. The similarity in this space is defined as:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq m}(1 + ||y_k - y_m||^2)^{-1}} \tag{2.8}$$

Concretely, firstly the algorithm calculates the probability distribution **P** for each samples, then it maps these values into a similarity probability **Q** in the lower space dimension. At the end it minimizes the Kullback-Leibler divergence between the distribution P and Q.

$$D_{KL}(P||Q) = \sum_{i \neq j} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{2.9}$$

To perform the optimization phase the algorithm exploits gradient descend. The parameters of the algorithm are:

- **The perplexity:** the number of neighbours to consider, this number can be seen as a trade off between the local and the global distribution of the samples
- **The learning rate**

t-SNE is implemented in scikit-learn.

2.4 Neural Network

2.4.1 Single Layer Perceptron

The idea behind perceptron [25] [27] comes directly from the nervous system, in fact we can see it as an artificial neuron. As in the analogy of the nervous system, a neuron i is connected with many other neurons and each of them propagates a signal. All the signals arrive at the neuron i and are summed together, and if the total balance is above a given threshold the neuron i spikes, this means that also itself propagate a signal, otherwise the neuron remain freeze. This is called all-or-none law.
The same occurs with the perceptron: it receives signals in input as the weights w, if the sum of all of them is above a threshold given b then the output of the perceptron will be 1, 0 otherwise.
Concretely it is a simple supervised algorithm for binary classification.

The formula that explains its behaviour can be written in this way:

$$f(x) = \begin{cases} 1 \quad if \quad w \cdot x + b > 0 \quad with \quad x_i = 0{,}1 \\ 0 \quad otherwise \end{cases} \tag{2.10}$$

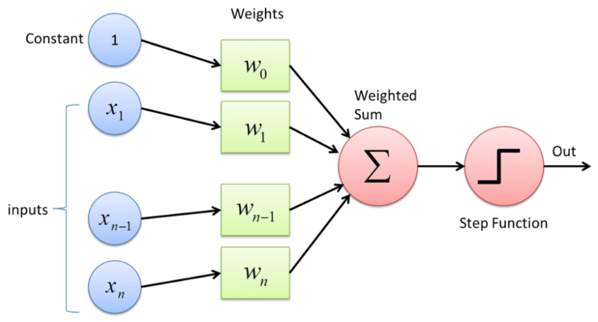


Figure 2.4: Perceptron [28]

With x_i = 0,1 means deactivate or active the output associated to w_i.

This can be also explained considering the perceptron as a weighted linear combination, followed by an Heaviside function.

$$f(x) = \sigma(\langle w, x\rangle + b) \tag{2.11}$$

The Heaviside in this case (but in general any not linear operator after a weighted linear combination) is called 'activation function' ie, a function that maps the result of the perceptron into another interval. Activation functions are a family of non linear function and add a non linear part to our computation.

At the end, this algorithm is simply a linear separating hyperplane that learns the right combination of weights in the training phase.

The rules for the weights update is the following:

Algorithm 1 Perceptron Algorithm

```
Require: w=0 and b=0
  repeat
    if $y_i[\langle w_i, x\rangle + b] < 0$ then
      $w \leftarrow w + y_i w_i \quad and \quad b \leftarrow b + y_i$
    end if
  until All classified correctly
```

2.4.2 Multi layer perceptron or Artificial Neural Network

Returning at the analogy with the nervous system, we can see an artificial neural network (ANN) [27] as a series of perceptrons that interact with each other!
An ANN is structured in layers of perceptrons, there are 3 types of layers:

- Input layer
- Hidden layers
- Output layers

Usually an ANN is composed by an input layer which is the input data, an output layer that that has as many perceptron as the classification classes and n hidden layers which play the role of features detectors.
Looking at the figure 2.5 we have 3 hidden layers, as seen in this scenario the the output of a neuron is the input for another one, and links represent different weights. If each neuron in a layer is connected with all the neurons in the next

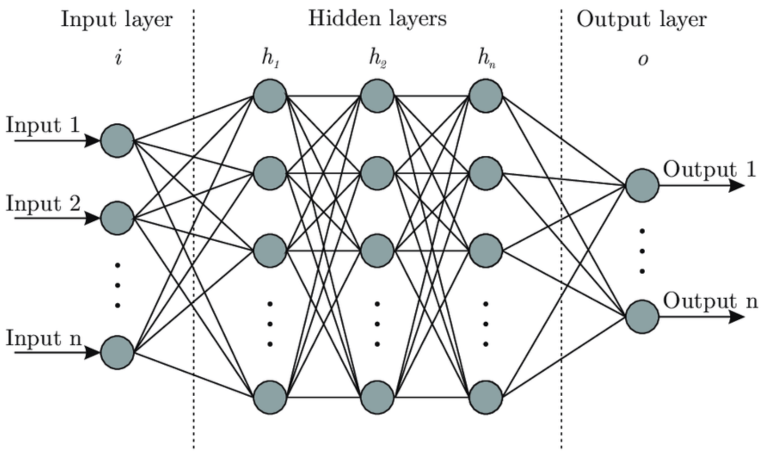


Figure 2.5: Artificial Neural Network [29]

layer the net is called fully connected. In this case, the input of a perceptron in a layer m is a linear combination of all the perceptrons in the layer $m - 1$.
After each perceptron it is needed an activation function, besides the Heaviside function (or Step function) we can find:

- **reLu (Rectified Linear Unit):** $f(x) = max(0, x)$
- **Sigmoid:** $f(x) = \frac{e^x}{1+e^x}$
- **Softmax**: $f(x) = \frac{e^x}{\sum_{j \in layer} e^j}$
- **Hyperbolic tangent**: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- **Leaky ReLu**: $f(x) = \begin{cases} 0.01x \quad if \quad x < 0 \\ x \quad if \quad x \geq 0 \end{cases}$

2.4.3 Training Phase

The procedure [27] for which the data in input are propagate from the input nodes to the output nodes is called **forward propagation**, and it has the goal to predict the classes.
The aim of the neural network is to find the right combination of weights that minimize the error between the predicted labels and the true ones. The error is expressed as *loss function*:

- **Hinge Loss:** $l(y) = max(0{,}1 - t \cdot y)$ with t binary classification index
- **Cross Entropy Loss:** $loss(x, class) = -log(\frac{exp(x[class[)}{\sum_j exp(x[j])})$

The loss function is calculated for each element, and then sum together to retrieve the error.
The aim is to find the right combination of weights that minimize the loss function. To do so we need to start with a random initialization of w, then predict the labels and calculate the loss.

The aim of the network is to minimize the error between the predicted labels and the true labels, ie the loss function. In other words, we need to find the right configuration of weights that minimize it.
To so do we should look the loss as a function of the weights, We should calculate how much each input is responsible of the error.
Looking at the figure 2.6 the red line represents the invert of the forward propagation, the error is propagating from the output through the network to the inputs.

The weights are updated based on the gradient of the Loss (so it is important for the loss function to be differentiable): ∇L_{w_i}, the gradient is calculated one gradient at a time and to go deeper into the network we utilize the chain rule for the partial derivative in order to arrive to the input layer.
This approach should be work together with an optimization method for the gradient descend.

We should adjust the weights on the direction that minimizes the gradient. This approach has a big problem since the gradient vanishes passing through the non linear activation functions, and it is more expressed as the network is deeper.

Usually to descend the gradient and update the weights the following optimization algorithms are used:

- **Gradient descend:**
  iterative optimization algorithm that should find the minimum

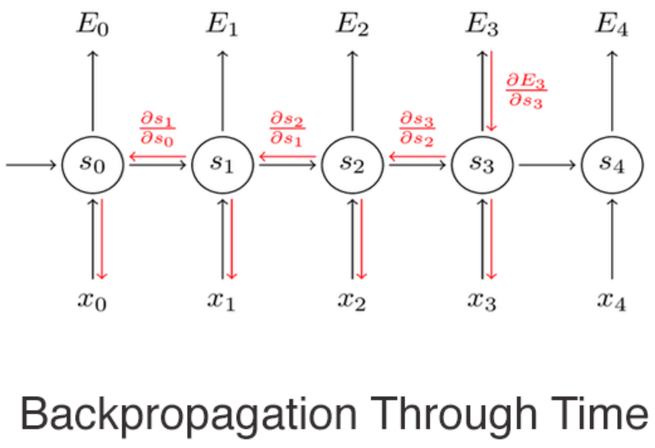

E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4
Backpropagation Through Time

Figure 2.6: Backpropagation

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

we moving a 'step' η in the opposite direction of the gradient in order to be near to the local minimum

- **Stochastic gradient descend**
- **Adam:**
  Extension of stochastic gradient descend with adaptive moment estimation.

Each algorithm has some hyperparameters to optimize:

- Learning rate
- Weight Decay
- Batch size

Also the number of hidden layers is a parameter that needs to be tuned.

2.4.4 Recurrent Neural Network (RNN)

Recurrent Neural Network or RNN [27] is a particular type of ANN which is able to preserve the temporal information of the data in input, taking as input sequences of samples rather than a single one.
For this reason RNN are a good instrument to study temporal series or Natural

Languages Problems (NLP).

We can visualize the architecture as the following: x_t the first data arrives at the block which performs some operation (schematized as the block in figure **??**) and returns an output, then this procedure is repeated recursively using each time as input the output of the previous time step. The block is basically an application of weights on the input data.

$$h_t = f_t(h_{t-1}, x_t) \tag{2.12}$$

Here a simple representation of the simplest RNN (Vanilla):

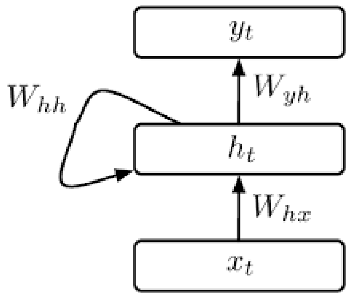


Figure 2.7: RNN Architecture [30]

$$\begin{aligned} h_t &= f_t(h_{t-1}, x_t) = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ y_t &= W_{hy}h_t \end{aligned} \tag{2.13}$$

By now , we saw example of 'one to one' architecture but for the purpose of our analysis is better to focus our attention to the 'Many to one' architecture.

In figure 2.8 we have 4 samples, that arrive consequently at the neural network, the firs sample x_1 is used to fed the fist block h_1, then the output of this operation is used as input for the second block together with the second input x_2 and so on. At the end we have just an output o_4.

It is important to notice that the blocks h_i have all the same composition in terms of weights.
Looking at the back-propagation phase, as mentioned before we risk that the gradient vanishes as the deepness of the network increases!

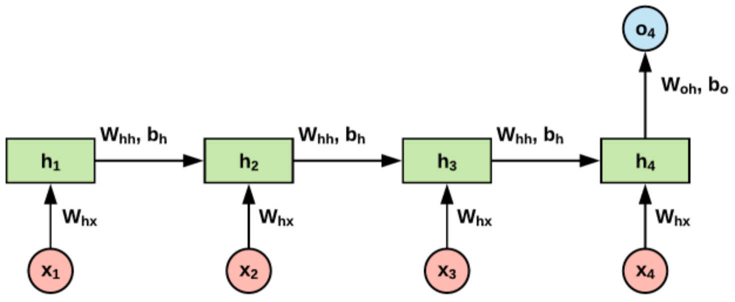


Figure 2.8: RNN [31]

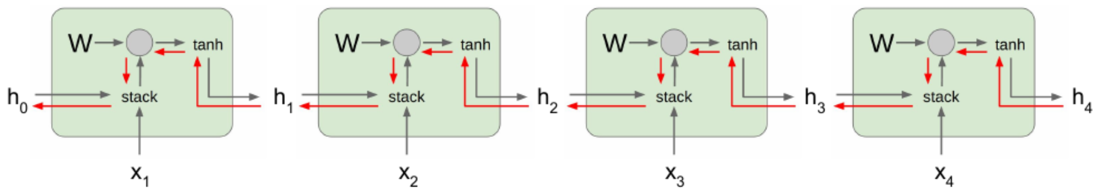


Figure 2.9: RNN Architecture [32]

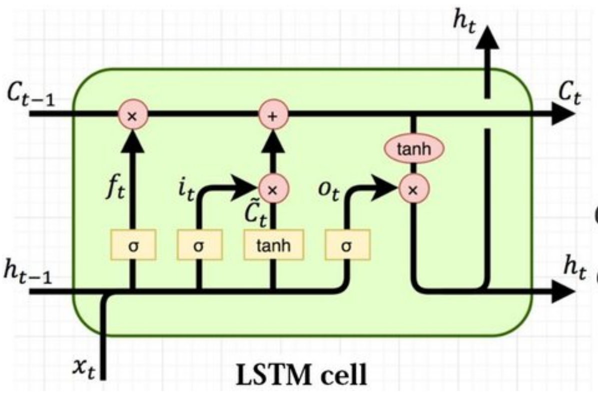
In fact, the derivative of *tanh* is 0 for its tails, so in these cases the gradient is 0! To overcome this problem, we should exploit Long Short Term Memory Network.

2.4.5 Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) is a type or RNN, which overcomes the vanishing gradient problem and so the error is free to back-propagate.
The architecture is formed by a series of units each one characterized by:

- Input gate i_t
- Output gate o_t
- Forget gate f_t

For each cell, the x_t is the input that is summed with h_{t-1} which is the hidden layer of the previous step.
C_t is passing through the network, f_t controls how much the network can forget the previous state and the u_t how much weight the input state.
With $h_0 = 0$ and $C_0 = 0$ As seen in figure 2.10, the C line is not under the control of any *tanh* so the gradient will not vanish!



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 2.10: LSTM [33]

As seen, i_t, f_t, o_t and $\tilde{C}_t$ are weighted linear combination of x_t and h_{t-1} following by an activation function.
As with the RNN, here can have a 'many to one' architecture, in the sense that we consider only the last h_t as output.

2.5 Embedding

The neural network cannot be fed with raw sequences of character, for this reason we need to implement some encoding technique in order to have the data in more suitable form to proceed with the deep learning analysis.

2.5.1 Word2Vec

Word2Vec [34] is an embedding technique mostly exploit in Natural Language Processing (NLP), that aims to learns the words association from a given corpus. Basically, it transforms each word into a numerical array exploiting a 2 layers neural network that should be able to extract the context, semantic and syntactic similarity from each word.

The aim is to create a system of dependencies among words, in the sense that words with similar meaning should be near each others in the embedding space.
We concentrate our study of the Common Bag Of Words (CBOW) version of this algorithm.
Imagining of having as dataset composed by a list of sentences. The first thing to do is to divided each sentence into words: this new representation will be our corpus.
Now each different word needs to be encoded into a one hot array, here for example:

"The dog chases the cat"
the = [1,0,0,0] *dog* = [0,1,0,0] *cat* = [0,0,0,1]
Each words is now represents by a vector with length equal to the number of different words = k, that contains all zeros instead of for a 1 at the position of the word.
For example, taking the word "chases" : the neural network should return as output the probability that each other words are in a window *d* from this word.

So the output vector is a k length array which express a probability for each word to be in the neighborhood of the input.

The algorithm is already implement in 'models', and takes as input:

- The corpus
- The dimension of the windows
- The dimension of the hidden layer

The training phase is performed on the corpus.
Here, the representation of each word is just the weights in the hidden layer.

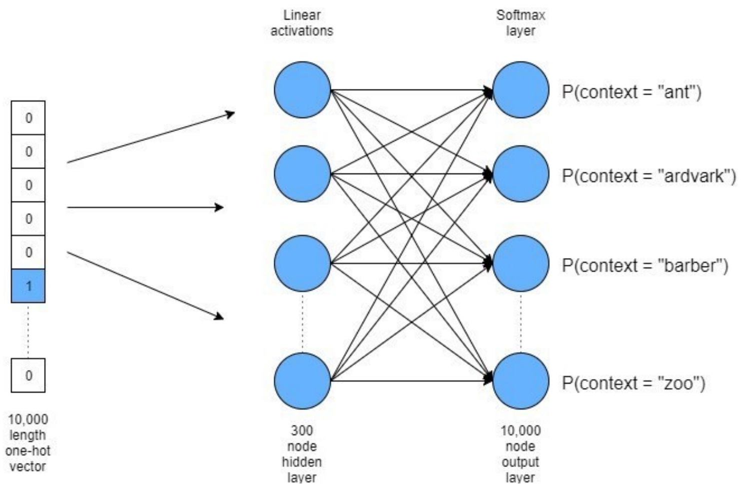


Figure 2.11: Word2Vec ANN [35]

Chapter 3

Sars-CoV-19 Spike Protein Analysis

3.1 Dataset

The data under analysis are collected from GISAID [36], which is an open-source website that provides nucleotides and aminoacids sequences of coronaviruses for many countries, together with analysis of the lineages and the variants of concerns.

3.1.1 Extraction

I downloaded a fasta file which contains the amino acid sequence of all the samples collected from Gisaid for different country in the world.
An element in the fasta corresponds of two lines with the following form:

$> Spike|hCoV-19/England/ALDP-959D26/2020|2020-06-11|$
$EPI\_ISL\_555111|Original|hCoV-19$^^*England/Human/Lighthouse Lab in Alderley Park/Wellcome Sanger Institute for the COVID-19 Genomics UK (COG-UK) consortium/Davies/United Kingdom*

"MFVFLVLLPLVSSQCVNLTTRT.....SCCKFDEDDSEPVLKGVKLHYT"

The first line contains the information about the sample such as date, laboratory, etc Besides, the second line represents the aminoacid sequence of the sampled spike protein.
Furthermore, the fasta file contains samples for different countries so the first step is to filter this file including only samples collected in England and store the contents

useful for the analysis in a csv file.

The resulting csv has the following form:

- **Epi ISL:** unique for each sample
- **Origin Lab:** laboratory where the sequencing is performed
- **Date:** date of the sample
- **Sequence:** list of amino acids of the spike protein

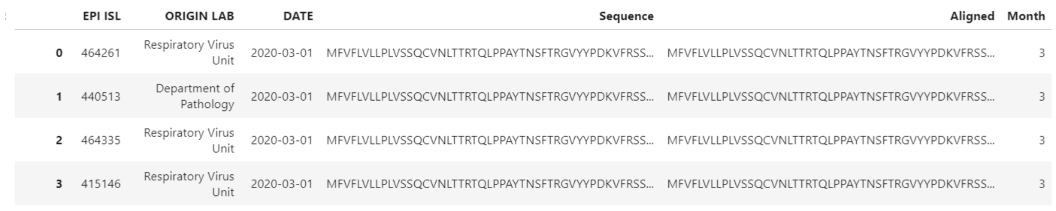
At the end of this procedure we collected 646697 samples from the England dataset.

3.1.2 Dataset cleaning

Furthermore, the collected dataset shows some badly reconstructed sequences where exist several ambiguous amino acids (i.e. not unequivocally identified) marked with a **X**. These samples have been removed from the dataset along with sequences with length lower than 1250 since also here we have an high occurency of badly reconstructed sequences, Moreover samples from January and February 2020 are discarded since they are too few. So, at the end I preserved only 461122 samples with a cleaning efficiency of 75%

3.1.3 Aggregation and Alignment

To perform the analysis firstly the dataset has been splitted in monthly bins in chronological order starting from March 2020 which is labelled as month 3 and arriving to August 2021 which is labelled as 20.
Then in order to be able to easily recover all the possible mutations each sequence have been pairwise aligned using as reference the original sequence from Wuhan.
At the end of this this procedure the data are represented as shown in the following figure:



	EPI ISL	ORIGIN LAB	DATE	Sequence	Aligned	Month
0	464261	Respiratory Virus Unit	2020-03-01	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	3
1	440513	Department of Pathology	2020-03-01	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	3
2	464335	Respiratory Virus Unit	2020-03-01	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	3
3	415146	Respiratory Virus Unit	2020-03-01	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...	3

3.2 Sequence Analysis

3.2.1 Dataset time dependence

In figure 3.1 we can see the overall daily cases of recorded sequences in England.The time dependence is of course fully correlated to the development of the three main pandemic waves.

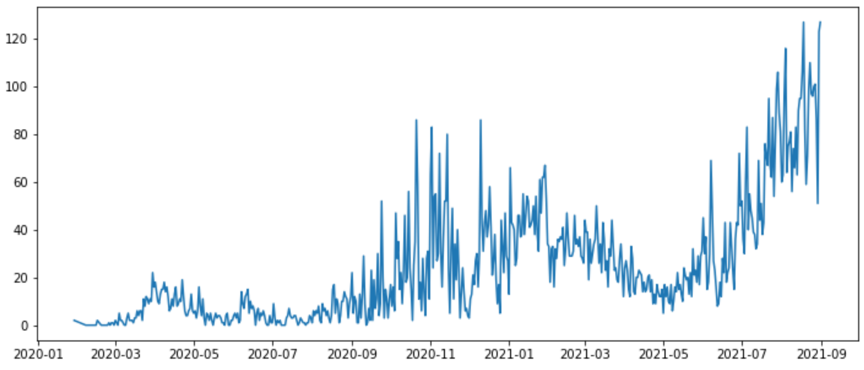


Figure 3.1: Daily swamps in England

Moreover, roughly speaking in fig.3.2 we can plot the daily cases taking into account also the length of the spike protein. This way a first rough separation of the main variants is clearly visible. Indeed the original spike protein has a length of 1273 while the alpha and delta variant have 1270 and 1271 sequence length. Notice that this separation is based only on the sequence length and the green tail in fig.3.2 should not be interpreted as an hint of an early presence of the delta variant.

3.2.2 Sequences duplication rate

Looking at the dataset the first thing to come at eye is that in each month we found sequences with a high duplication rate (see tab.3.1).

We should take into account this high repetition rate through all the analysis, indeed it is biologically coherent that only few sequences are strongly repeated and around them some mutations occur. Besides, this should lead some problems in terms of computation.

Moreover as we can see in Figure 3.3 the overall rate of different sequences is almost flat.

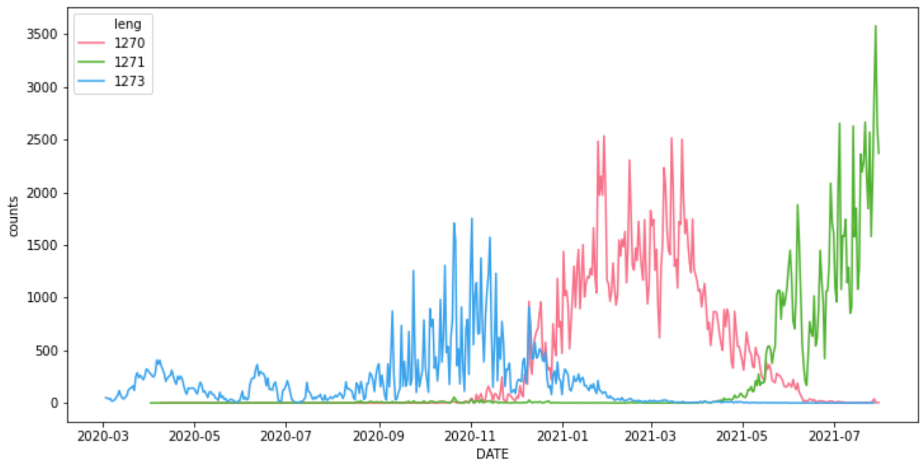


Figure 3.2: Daily swamps by length

Month	Total Sequences	Different Sequences
3	3932	172
4	7315	405
5	2195	208
6	4904	282
7	1960	163
8	4289	317
9	9412	533
10	19121	1071
11	23290	1500
12	25278	1435
13	46489	2045
14	38848	1697
15	49541	1780
16	24053	1187
17	24328	953
18	30981	1257
19	58600	2540
20	86520	3896

Table 3.1: Number of total and different sequences for the England dataset

Furthermore, in Figure 3.4 it is noticeable to see the number of sequences that are highly repeated compared with the ones that are less repeated.

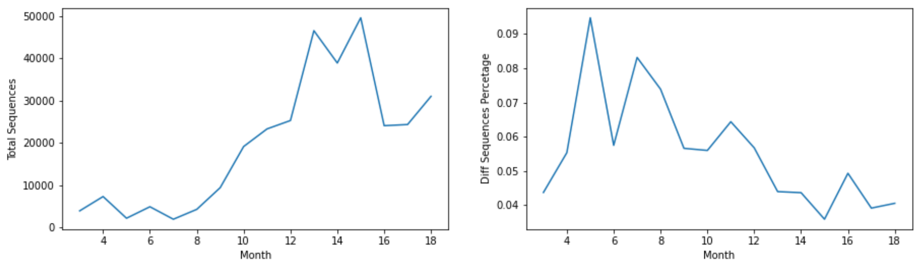


Figure 3.3: Number of total and different sequences

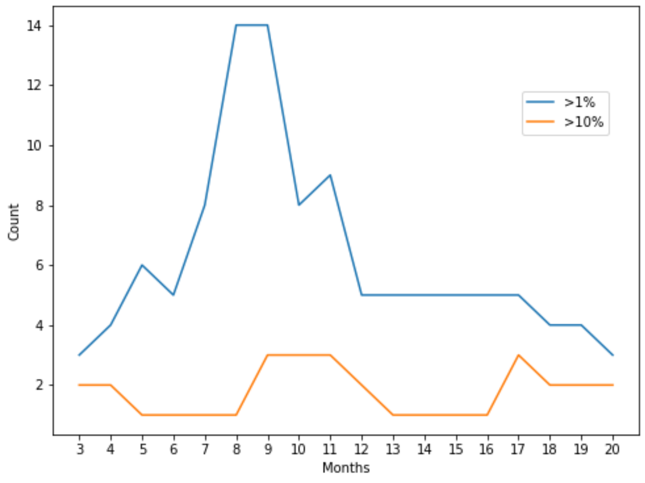


Figure 3.4: Sequence Percentage

3.2.3 Spike protein overview

At the end, it is interesting to notice how much the protein can change. In fact, mutations are allowed only in particular sites, the protein is allowed to change but not to modify itself too much, otherwise it will lose its biological properties.
At the end, it is useful to remember that the mutations that are more biologically advantageous are the ones that improve the transmission. In fact, a virus to spread must be easily infectious, should multiply itself rapidly and should not be too lethal, otherwise it kills the host and does not pass the mutation to other ones.
To easily visualize the possible hot-spots of the mutations the number of different aminoacids we found in each site (a sequence is 1273 amino acid long) is shown in Figure 3.5.

Some regions shows higher mutation rate, in particular at the beginning of the

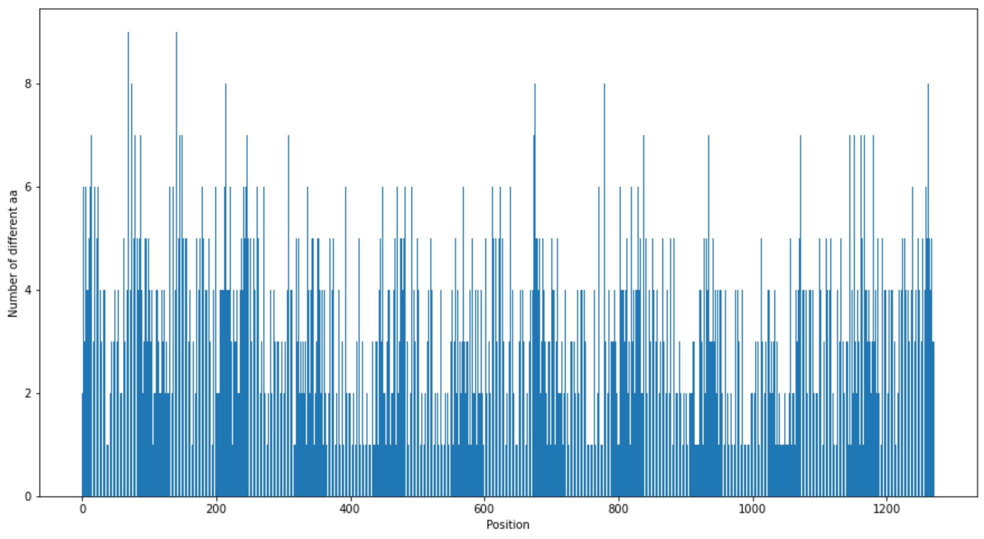


Figure 3.5: Mutations global hot-spots

sequence. Indeed this region contains the Region Binding Site that plays a key role in the virus interaction with the host cell, so a mutation in this region is more likely to lead to a significant biological change.

3.3 Cluster Analysis

The aim of this part is to divide the sequences in each time step (here month) into different groups based on their similarity. In fact, we want similar sequences to be grouped together.

3.3.1 Distance Matrices

To have a measure of the pairwise similarity among sequences we build for each month a matrix that contains the pairwise Levensthein distance among all the sequences in a month.
This is a square matrix in which each row and column represent a sequence, the intersection between a row and a column is the distance between these 2 sequences. This step is really time consuming, so firstly it is built only a matrix containing the distance among different sequences, after this the matrix is expanded looking at the multiplicity of each sequence.

3.3.2 Clustering procedure

Hierarchical clustering is performed monthly based on Ward distance. A common technique is needed in order to be coherent along months, so the same parameters should be applied for all the steps.

The clustering procedure is described in the following lines:
Firstly the hierarchical clustering is performed, then we look at the number of elements in each clusters. We don't want to have too many clusters or clusters with few samples. So a dedicated study has been made to define a cut for the minimum number of elements that should have a cluster in order to be taken in account.However the elements that are discarded are then reassigned to the nearest cluster above the threshold.

Schematically:

1. Hierarchical Clustering
2. Check the number of sequences in each cluster
3. If the cluster is below the threshold then we re-assign the sequences to the nearest valid cluster

To perform the reassignment we take one by one the sequences that are still free. For each sequence we calculate its average distance with all the sequences in each cluster. At the end, the sequence is assign to the cluster with the minimum distance.

So this procedure require the simultaneous optimization of two parameters:

- **Threshold:** the distance at which cutting the tree of the hierarchical clustering
- **Cut:** the minimum number of element needed in a cluster to be selected

The Threshold parameter impact on the number of clusters defined while the cut parameter on the number of sequences we need to reassign. In figure 3.6, we can see an example of setting the threshold parameter at *th=100* and the cut parameter to *cut=1% of the dataset.* We can see that we find 4 clusters but only 2 of them are above the minimum number of samples needed to be defined. The discarded clusters are then reassigned to the nearest one, which is this case is the red cluster.

3.3.3 Clustering procedure: Working point optimization

The aim of this part is to find an unbiased criteria to optimize the *'working point'* for the two parameters described in sec.3.3.2

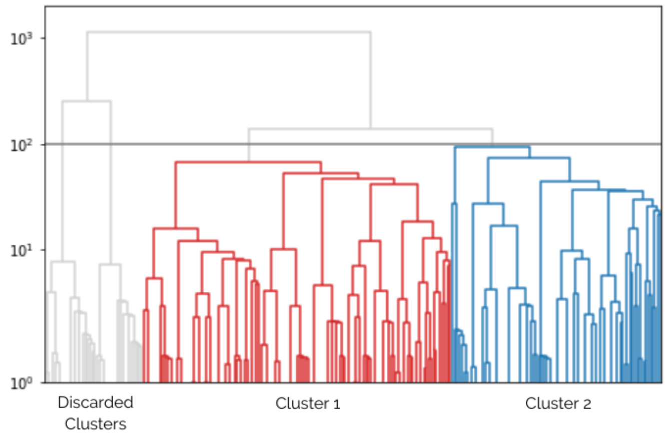


Figure 3.6: Distance tree and clusterization (month 9)

Here we can evaluate the impact of the parameters choice on:

- **Percentage** of dataset grouped in a cluster below the threshold
- **Number of clusters** (we would like to keep this number under control)

To find the optimal working points we looked at the cut parameters and we found cut=1% = 10^-2 of the total elements in the month as a reasonable value. After that once we set the cut value 1% = 10^-2 we studied the two main variables (namely the percentage of dataset grouped in a cluster and the number of clusters) for different values of the threshold parameter.

As seen in the figures 3.7 and 3.8 after threshold=10^2 the system reach a quite stable plateau both for the percentage of dataset grouped in a cluster and the number of cluster. Moreover, at this threshold the number of clusters are enough to be specific but at the same time enough low to allow an investigation in terms of reasonable computational time.

At the end of this optimization we set:

- **Threshold=100**
- **Cut=** 10^{-2}

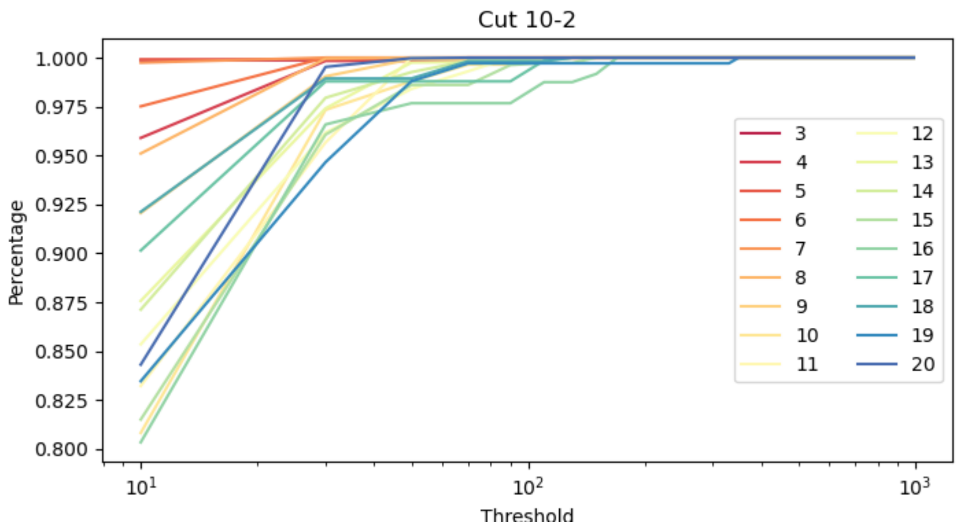


Figure 3.7: Elements not discarded/total elements

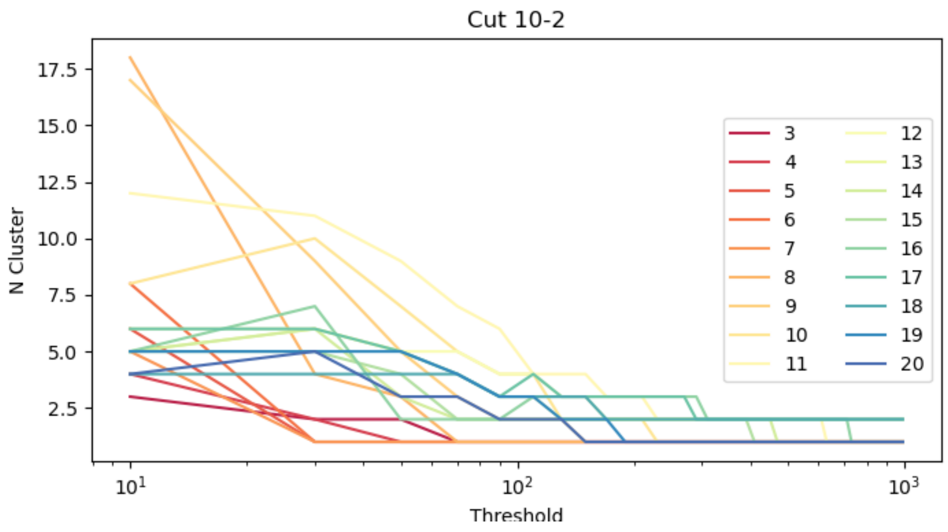


Figure 3.8: Number of Clusters

3.3.4 Clustering procedure: Results

3.3.2 After the optimization of the working points we applied the clusterization procedure and the results are reported in the following lines. Clusters have been enumerate by crescent order. For each of them we reported the percentage wrt the monthly dataset and the number of different sequences.

In month 3
Cluster number 1 represents 1.0 of the dataset, with 172 different sequences, max difference: 21.0

In month 4
Cluster number 2 represents 1.0 of the dataset, with 398 different sequences, max difference: 5.0

In month 5
Cluster number 3 represents 1.0 of the dataset, with 208 different sequences, max difference: 23.0

In month 6
Cluster number 4 represents 1.0 of the dataset, with 282 different sequences, max difference: 11.0

In month 7
Cluster number 5 represents 1.0 of the dataset, with 163 different sequences, max difference: 5.0

In month 8
Cluster number 6 represents 1.0 of the dataset, with 317 different sequences, max difference: 24.0

In month 9
Cluster number 7 represents 0.64 of the dataset, with 348 different sequences, max difference: 23.0
Cluster number 8 represents 0.36 of the dataset, with 185 different sequences, max difference: 6.0

In month 10
Cluster number 9 represents 0.34 of the dataset, with 335 different sequences, max difference: 10.0
Cluster number 10 represents 0.27 of the dataset, with 225 different sequences, max difference: 6.0
Cluster number 11 represents 0.37 of the dataset, with 479 different sequences, max difference: 7.0
Cluster number 12 represents 0.02 of the dataset, with 32 different sequences, max difference: 12.0

In month 11

Cluster number 13 represents 0.35 of the dataset, with 416 different sequences, max difference: 18.0
Cluster number 14 represents 0.08 of the dataset, with 57 different sequences, max difference: 15.0
Cluster number 15 represents 0.28 of the dataset, with 426 different sequences, max difference: 6.0
Cluster number 16 represents 0.23 of the dataset, with 500 different sequences, max difference: 9.0
Cluster number 17 represents 0.05 of the dataset, with 73 different sequences, max difference: 9.0
Cluster number 18 represents 0.01 of the dataset, with 28 different sequences, max difference: 9.0

In month 12
Cluster number 19 represents 0.2 of the dataset, with 336 different sequences, max difference: 21.0
Cluster number 20 represents 0.56 of the dataset, with 485 different sequences, max difference: 13.0
Cluster number 21 represents 0.15 of the dataset, with 296 different sequences, max difference: 7.0
Cluster number 22 represents 0.08 of the dataset, with 318 different sequences, max difference: 12.0

In month 13
Cluster number 23 represents 0.88 of the dataset, with 1403 different sequences, max difference: 15.0
Cluster number 24 represents 0.1 of the dataset, with 436 different sequences, max difference: 17.0
Cluster number 25 represents 0.02 of the dataset, with 202 different sequences, max difference: 18.0

In month 14
Cluster number 26 represents 0.98 of the dataset, with 1466 different sequences, max difference: 31.0
Cluster number 27 represents 0.02 of the dataset, with 221 different sequences, max difference: 14.0

In month 15
Cluster number 28 represents 0.99 of the dataset, with 1554 different sequences, max difference: 15.0
Cluster number 29 represents 0.01 of the dataset, with 176 different sequences, max

difference: 24.0

In month 16
Cluster number 30 represents 0.95 of the dataset, with 1008 different sequences, max difference: 16.0
Cluster number 31 represents 0.03 of the dataset, with 50 different sequences, max difference: 22.0

In month 17
Cluster number 32 represents 0.42 of the dataset, with 519 different sequences, max difference: 24.0
Cluster number 33 represents 0.39 of the dataset, with 235 different sequences, max difference: 8.0
Cluster number 34 represents 0.18 of the dataset, with 144 different sequences, max difference: 10.0

In month 18
Cluster number 35 represents 0.73 of the dataset, with 793 different sequences, max difference: 14.0
Cluster number 36 represents 0.21 of the dataset, with 267 different sequences, max difference: 8.0
Cluster number 37 represents 0.06 of the dataset, with 175 different sequences, max difference: 8.0

In month 19
Cluster number 38 represents 0.8 of the dataset, with 1923 different sequences, max difference: 18.0
Cluster number 39 represents 0.15 of the dataset, with 416 different sequences, max difference: 11.0
Cluster number 40 represents 0.04 of the dataset, with 149 different sequences, max difference: 7.0

In month 20
Cluster number 41 represents 0.84 of the dataset, with 1271 different sequences, max difference: 15.0
Cluster number 42 represents 0.16 of the dataset, with 319 different sequences, max difference: 22.0

As stated before, the sequences are highly repeated so clusters in the same month differ a lot in terms of number of sequences. Besides, we can imagine a cluster having as centroid its most repeated sequence which represents the greater part of its elements. As, said before this is biologically acceptable.

3.4 Linkage Analysis

Now that each time step is divided into groups of sequences we can proceed linking together consecutive clusters to reconstruct the time pathway, and retrieve evolutionary dynamics. Hence, firstly we need to to define rules to connect together clusters that belong to consecutive months.
In figure 3.9, we can see the cluster diagram. Each color represents a different

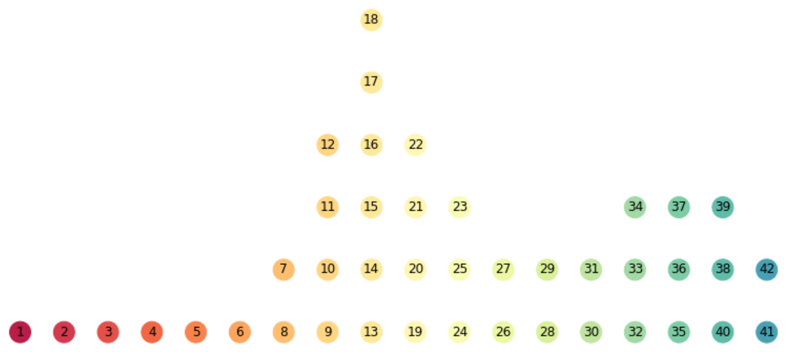


Figure 3.9: Clusters of sequences vs time.Each color represents a different month (time step is one month)

month, number are assigned to cluster as reported in sec. 3.3.2.
We can already see that since in the beginning the samples are very similar we only got one or a few clusters per month, and we only started seeing an increase of clusters in early fall 2020.

3.4.1 Strong Links

As expected, each cluster is defined by a dominant sequence, that identifies it uniquely.
The linking has been done looking for similar clusters forward in time. To do so we retrieved the dominant variant of each cluster and if clusters in consecutive month with the same dominant sequence have been found we connected them in a strong way, using black links.

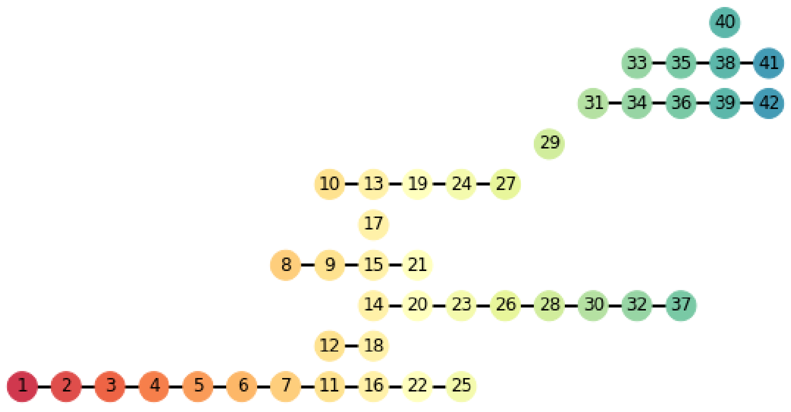


Figure 3.10: Strong links

Reordering the clusters in a more fancy way and plotting the strong links in Figure 3.10, we are already able to see some chains!

Taking into account that these clusters are characterized by the same dominant sequence we can see that each black chain is defined by its dominant variant which is unique.
Moreover, for each chain we can find some correspondences between the dominant sequences and variants in the Pangolin lineage [37] and variants are named as described in [38]. For each variant the mutations are expressed taking as reference the sequence from Wuhan. As an example let's take the string *D*614*G*: It means that the amino acid D (aspartic acid) in the original sequence is substituted by a G (glycine) in position 614.
Also for simplicity to each time-ordered cluster chain we associated a short name that will be used in the following discussions. Here we report the complete list of the time-ordered cluster chains:

- Chain [1,2,3,4,5,6,7,11,16,22,25] corresponds to B.1.1.1:
  It is the variant that was spreading in Europe in the beginning of 2020.
  Mutations: D614G
  For simplicity: v0
- Chain [8,9,15,21] corresponds to B.1.177:
  It is one of the variant responsible for the second wave in the UK
  Mutations: D614G, A222V
  For simplicity: v1a

- Chain [10,13,19,24,27] corresponds to B.1.177 + L18F:
  It is one of the variant responsible for the second wave in the UK
  Mutations: D614G, A222V, L18F
  For simplicity: v1b

- Chain [12,18]:
  As we will see, this variant contains one of the main mutation in the Alpha Variant.
  Mutations: H69-, V70-, N439K, D614G

- Chain [14,20,23,26,28,30,32,37] it corresponds to the B.1.1.7:
  Alpha Variant, the main responsible for the second wave in the UK.
  Mutations: H69-, V70-, Y144-, N501Y, A570D, D614G, P681H, T716I, S982A, D1118H
  For simplicity: v2

- Chain [31,34,36,39,42] corresponds to the B.1.617.2 :
  Delta variant is causing the third wave
  Mutations: T19R, G142D, E156-, F157-, R158G, L452R, T478K, D614G, P681R, D950N
  For simplicity: v3a

- Chain [33,35,38,41] corresponds to the B.1.617.2 + T95I:
  Delta variant plus 1 mutation is the main cause of the third wave
  Mutations: T19R, T95I, G142D, E156-, F157-, R158G, L452R, T478K, D614G, P681R, D950N
  For simplicity: v3b

For all these chains we calculated the average distance between sequences belonging to consecutive clusters. As shown in Figure 3.11 we found the larger part of strong links at very low distances values (100% of events at distance ≤ 3.5)

3.4.2 Weak Links

If there are clusters in consecutive months that we cannot connect via strong links then it is possible to inspect the average distance between the sequences in these clusters.
To do so, we need to calculate the average distance matrices. Since this time the dominant sequence for the two clusters is not the same we could expect larger values wrt the ones found for the strong links (that we saw before are always ≤ 5). This way, we can say such clusters are connected with weak links . In order to have a one-to-one architecture (to preserve the linearity of the chains) if we found

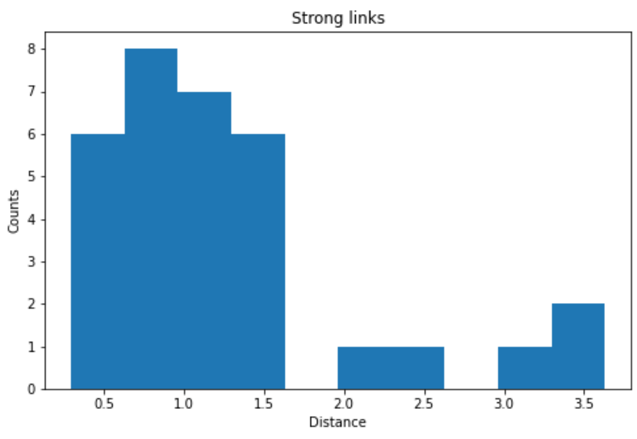


Figure 3.11: Strong Links distances

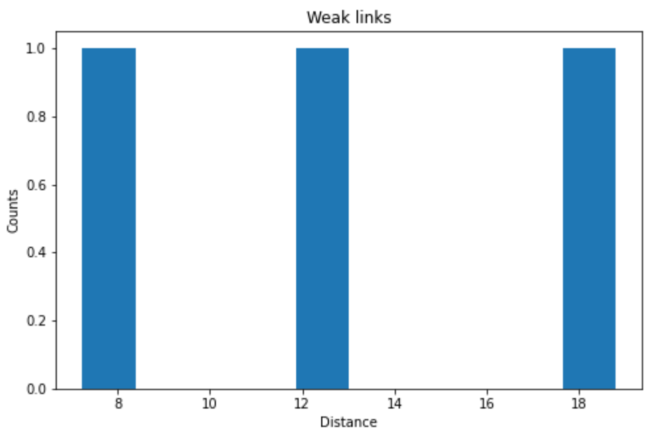


Figure 3.12: Weak Links distances

two weak links that point to the same cluster we kept the one that minimizes the average distance.

Anyway we found that using the working points **Threshold=** 100 and **Cut=** 10^{-2} defined in sec. 3.3.3 on the England dataset we were able to define all the time-ordered cluster chains **using only strong links**.

Besides, this procedure will be useful in the further sections.

3.4.3 Chains

A time ordered cluster chain is defined as a list of clusters linked together with strong links, so looking at fig. 3.10 we have:

- [1,2,3,4,5,6,7,11,16,22,25]
- [8,9,15,21]
- [10,13,19,24,27]
- [14,20,23,26,28,30,32,37]
- [31,34,36,39,42]
- [33,35,38,41]

3.4.4 Branching Links

At the end, it is needed to find where each time-ordered chain comes from!
To answer this question we analyze the sequences inside the first cluster of each chain and then we look for a possible correspondence in the previous month.

To do so we define the following branching algorithm:

Algorithm 2 Branching algorithm for find the parenthood of cluster i

```
n= month of cluster i
s = empty array
for All the cluster k in month n-1 do
    $s_k$= Number of sequence in common between cluster i and cluster k
end for
if s not empty then
    Look at the maximum value of $s_k$
    k is the parent of i
end if
```

Applying this algorithm for all the time-ordered cluster chain we finally retrieve the complete graph as shown in fig.3.13:

These branching links represent a way to inspect the origin of each chain. In fact, as shown in the algorithm definition, we want to see where the most of the sequences in our target comes from. This way We are able to investigate the history and the origin of each (dominant) variant.

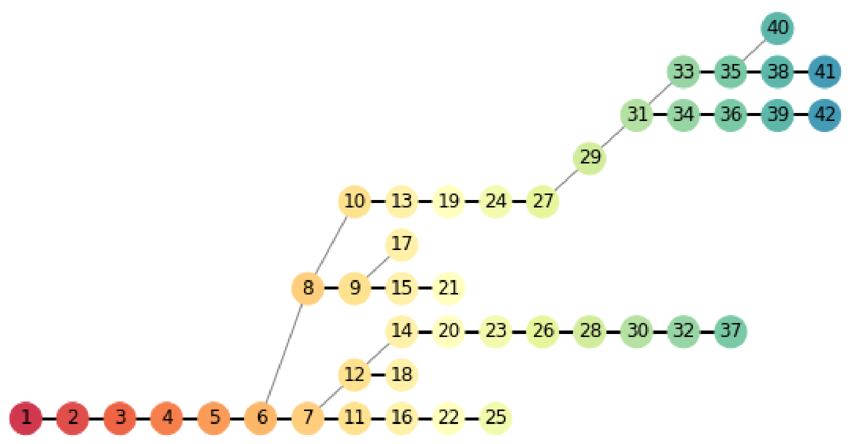


Figure 3.13: Linkages

3.4.5 Time ordered chain Analysis

Once we define the complete list of the time ordered cluster chain it has been possible to extract some useful statistics.

For the sake of simplicity we assign a color to each time ordered chain:

- v0: blue
- v1a: red
- v1b: pink
- v2: gold
- v3b: green

First of all it is possible to study the monthly percentage prevalence of each chain as shown in Figure 3.14. Here a competitive mechanism between different variants is quite evident. Indeed we can see that the blue line (associated to the v0) starts decreasing with the arrival of the "new" v1a and v1b variants (red and pinks one). Same way the v1's have been completely replaced by the alpha variant in a relatively short time window. A the end this last died with the increasing of the delta variant.

Furthermore, we can multiply these percentage with the number of total infected individuals in England. This way it is noticeable the composition of each pandemic waves in terms of variants of concerns (fig.3.15).

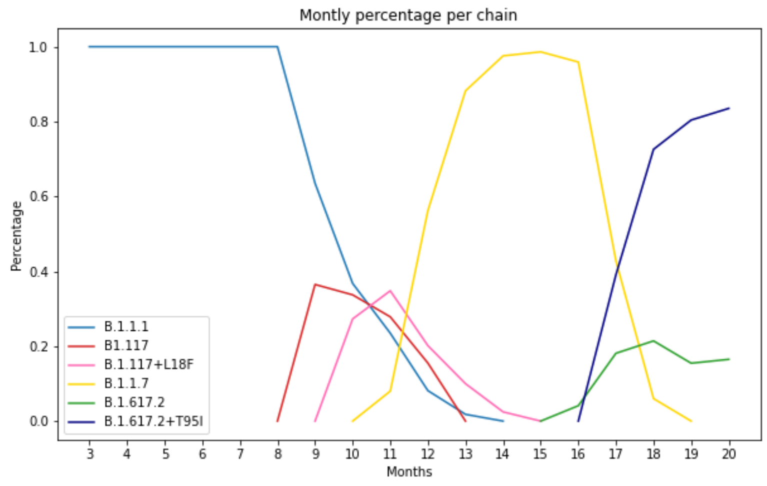


Figure 3.14: Monthly percentage

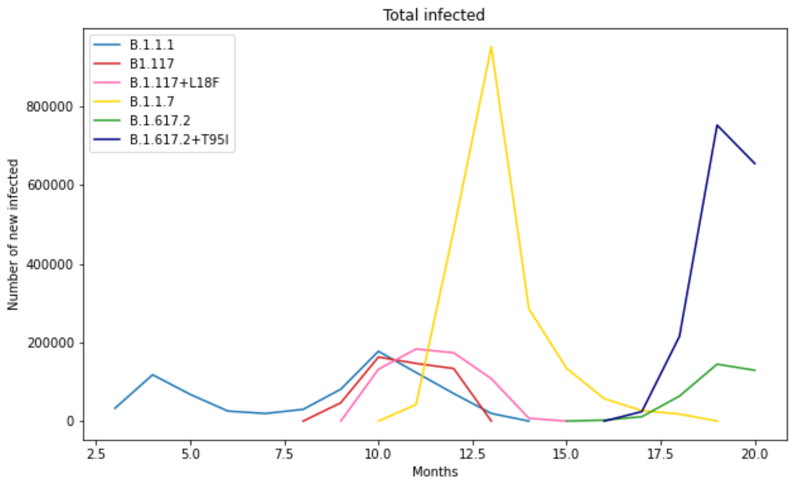


Figure 3.15: Pandemic wave composition

At the end, it is interesting to study how much clusters in the same chains are evolving through time. We calculated the average distance between consecutive clusters in the same way as we calculate the distance between clusters in the strong links case. Plotting these distances vs time (see Figure 3.16) we can have an hint of the chain time evolution. In particular the long lived chain (i.e. connected to dominant variants) show a quite evident increase but with some local decrease. This effect is due to the branching of a new chain that falls out from the original line, so the similarity of the overall remaining elements is increased. As an example in chain [13,17, *etc*] we can see that the distance decreases dramatically between cluster 35 and 38, this happens because a cluster is branching out from this chain and decreases the value of dissimilarity.

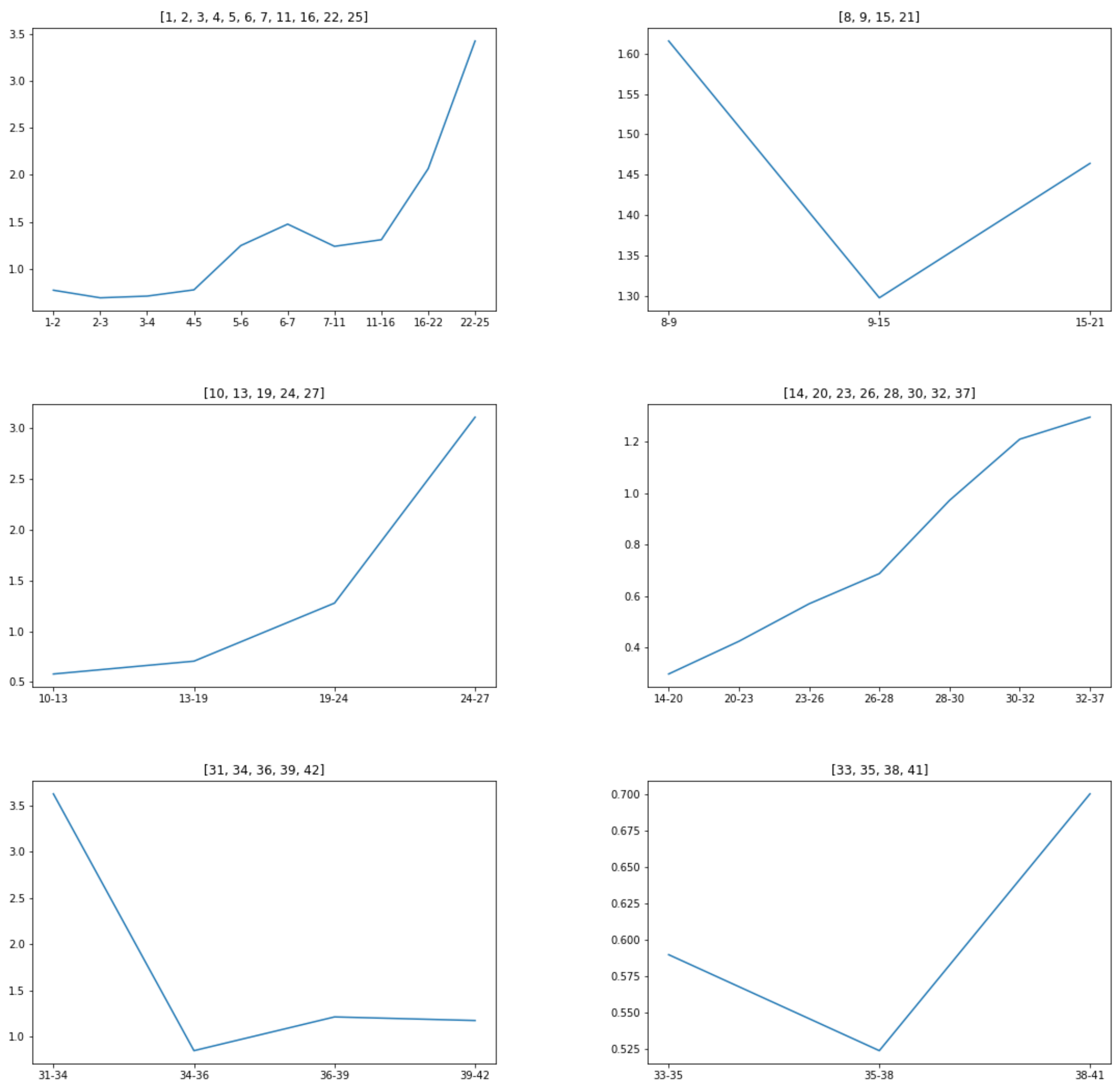


Figure 3.16: Time evolution of distances between clusters belonging to the same chain

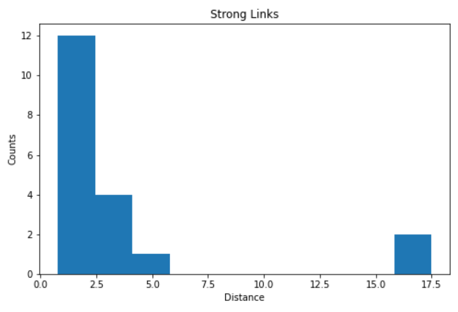
Moreover we can notice that the most stable chains show low values with reference to the more unstable chains.

3.4.6 A closer look to the alpha variant

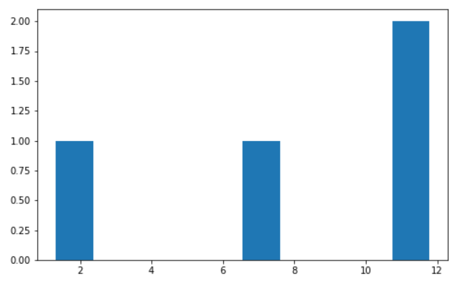
The alpha variant starts at node 14, and it is generated from node 12 (in fact inside node 12 we have found 70 sequences of the alpha variant) that is generated from node 7 (5 sequences of alpha variant) of the v0. Also, the deletions in 69-70 are antecedent the alpha variant and as said before they are contain in the most frequent sequence of cluster 12.
Also the dominant sequence is cluster 12 has one more mutation in N439K, that vanishes in the v2 chain. This because it finds a more suitable mutation to match the original one.

3.4.7 Validation with Threshold = 200

The same analysis is performed increasing the threshold of the hierarchical clustering to 200. This way we were able to agglomerate together the most similar clusters and having a simplified description of the variants spread. Strong and weak links distance distributions are shown in fig.3.17a and 3.17b.



(a) Strong link distance



(b) Weak link distance

Figure 3.17

Time orderd cluster chain results are shown in fig.3.18. Also in this case we can look at the dominant variant for each chain:

- [1,2,3,4,6,8,10,12,16]: v0, blue in the graph
- [11,14,17,20,22]: v1b, red
- [15,18,19,21,23,25,28,31]: v2, gold
- [16,29,32]: v3b, green

The monthly percentage prevalence of each chain is shown in Fig.3.19a while the composition of each pandemic waves in terms of variants of concerns is shown in Fig.3.19b.

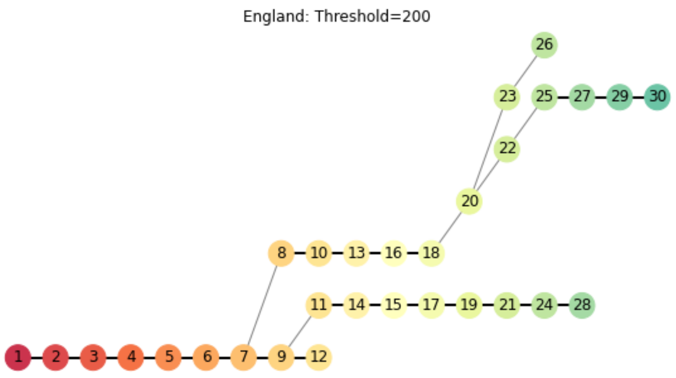
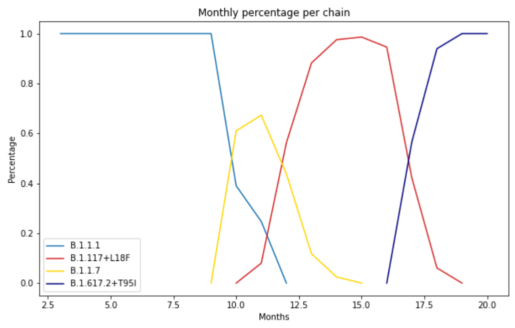
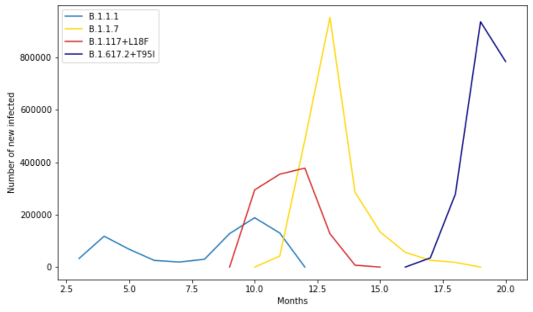


Figure 3.18: Time ordered cluster chains for threshold = 200 (England dataset).



(a) Monthly percentage



(b) Pandemic waves composition

Figure 3.19

3.4.8 Cross-Country validation

To validate our method, the same analysis is here performed for Wales and Scotland dataset. Although the dataset size is one order of magnitude smaller if compared with the England one, we can compare the results for Wales and Scotalnd wrt England to validate our method and also extract some useful information.

Time ordered cluster chains (Scotland) : Results for Scotland dataset are shown in fig.3.20. We were able to reconstruct three time ordered chains:

- [1,2,3,4,5,6,7,8,9,10,13,15] is associated with B.1.1.1 for the first part, and then it is associated to B.1.117+L18F
  In the graph: Blue

- [11,12,14,16,17,19,20] is associated with B.1.1.7
  In the graph: Yellow
- [18,20,22] is associated with B.1617.2+T19R
  In the graph: Green

Time ordered cluster chains (Wales) : Results for Wales dataset are shown in fig.3.21. Also here we were able to reconstruct three time ordered chains:

- [1,2,3,4,5,6,7,8,9,10,12,15,17] is associated with B.1.1.1 for the first part, and then it is associated to B.1.117+L18F
  In the graph: Blue
- [11,13,14,16,18,19,22] is associated with B.1.1.7
  In the graph: Yellow
- [20,21] is associated with B.1617.2
  In the graph: Green

The results are compatible with the previous analysis taking into account the statistical uncertainty which represent the main limitation of the dataset collected in these two countries where the number of recorded sequences is fairly less than the data collected in England. Anyhow, although this limitation, we were able to identify variant of concern in Wales and Scotland using the same approach we applied for the England dataset.

Using the results from England, Wales and Scotland dataset we can look at behaviour of the same time ordered chain in each of these countries. In particular, in figure 3.22 it is shown the comparison for the time evolution of the percentages (B.1.1.1 or v0: left plot, B.1.1.7 or v2: right plot).
The three curves look very similar (both vor v0 and v2) except for a time shift. In particular, we can appreciate how the B.1.1.1 percentage start to decreases 1-2 months before in England, and how B.1.1.7 arises a month before respect to Wales and Scotland.
This is in good agreement with the fact that the alpha variant was first recorded in the England dataset.

3.5 Early warning tool

The results for time-ordered chains with monthly clustering (fig. 3.13) show an almost perfect one-to-one correspondence with dominant variants but this choice put a stringent limit on the time resolution of the procedure, i.e. the minimum amount of time needed to isolate a new chain ($\simeq$ 1 month).

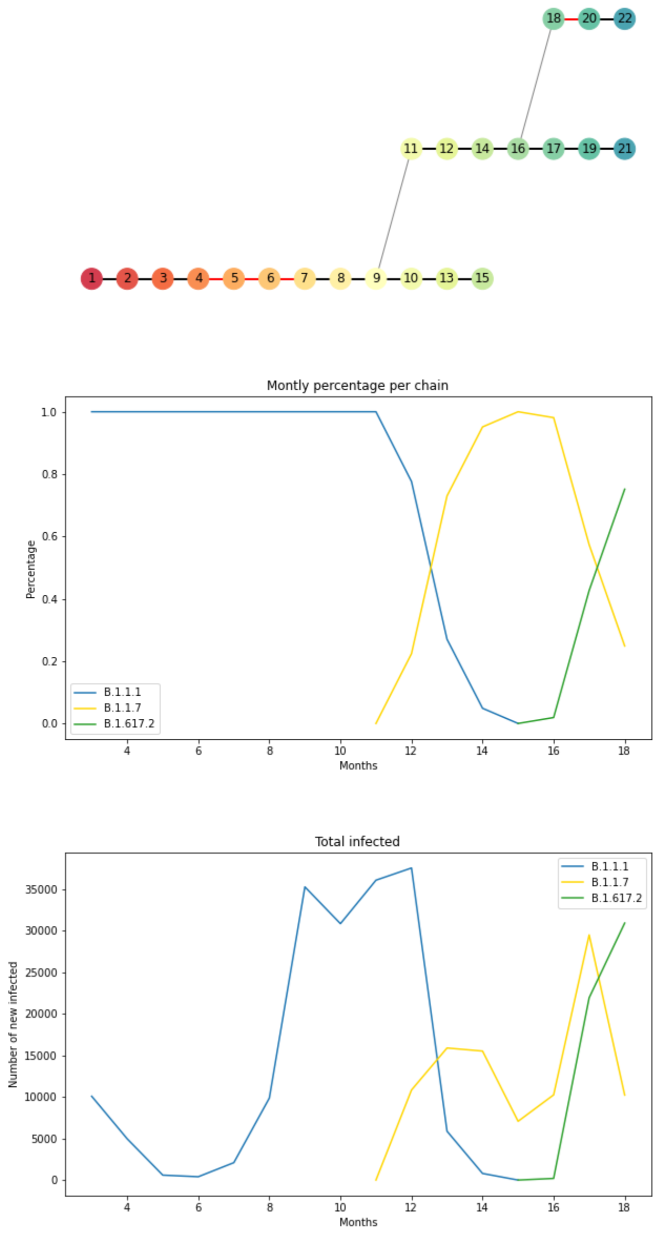


Figure 3.20: Scotland dataset results

Given the amount of data available for the England dataset, we can reduce the granularity of our analysis down to weekly bases. This way we can assess the reliability of our procedure as an early warning tool able to prompt react to new potentially dangerous variants.

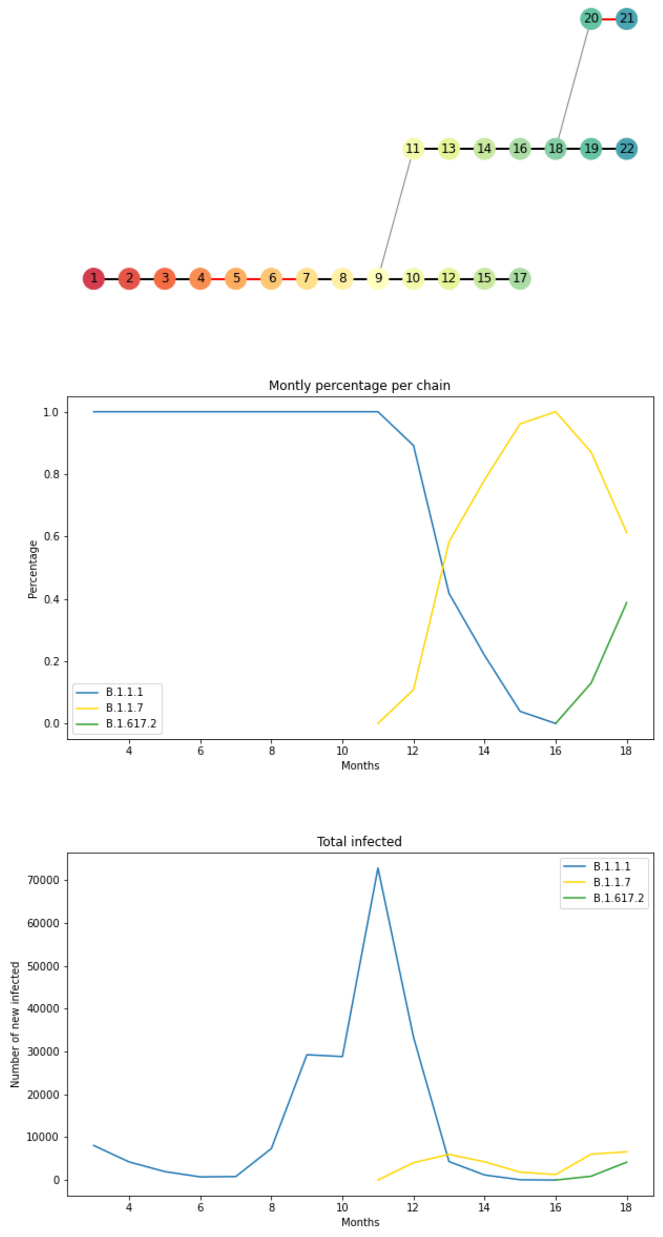
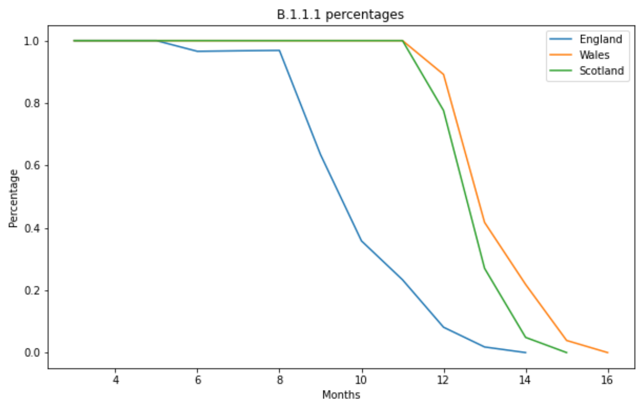


Figure 3.21: Wales dataset results

To do so, we firstly divided our dataset by weeks and recreated all the distance matrices based on this time scale, then the clusterization phase were performed once again. After this step we performed the study on the the alpha variant (v2)



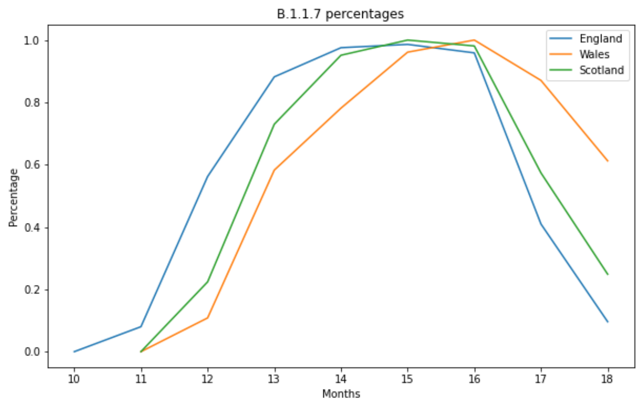


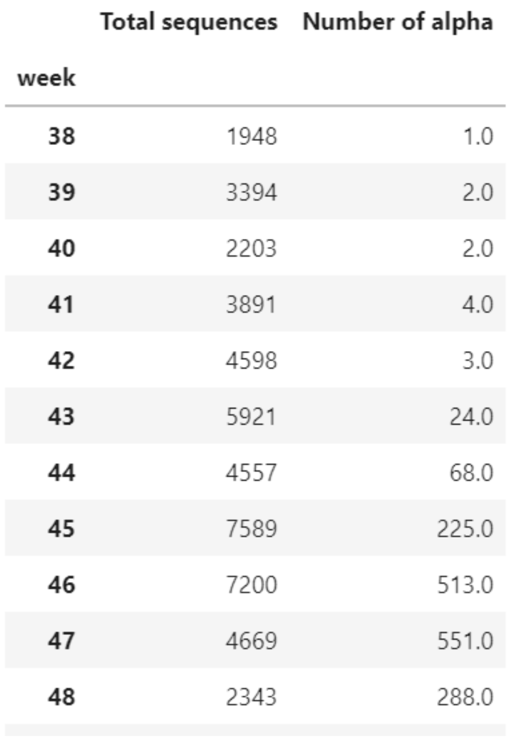
Figure 3.22: B.1.1.1 (v0) and B.1.1.7 (v2) time evolution comparison

case.

In our dataset, the first case of alpha variant is registered on 2021-09-20, during week 38 and then it stays at low values of cases for some weeks. In Figure 3.23 there are depicted the number of alpha variants we found in each week with the number of total sequences in that week.

In our monthly analysis we were able to successfully identify the alpha variant in November (weeks 45-46-47-48). So at the end we have to wait until the end of week 48 to correctly identify this variant.

Keeping the values of threshold and cut at the same values defined in the previous section (threshold=100 and cut=1%) we succeeded to find the alpha variant cluster first time in week 44, indeed as shown in the Figure 3.23 for all the previous weeks we have too few samples of alpha to satisfy the cluster cut value of 1%.



	Total sequences	Number of alpha
week		
38	1948	1.0
39	3394	2.0
40	2203	2.0
41	3891	4.0
42	4598	3.0
43	5921	24.0
44	4557	68.0
45	7589	225.0
46	7200	513.0
47	4669	551.0
48	2343	288.0

Figure 3.23: Alpha variant per week

However we can still modify the values for the two parameters threshold and cut to find the best values of these two parameters that enable us to see the alpha variant before week 44. This is possible and indeed has been successfully done performing a two dimensional grid search. Here the price to pay is the growing numbers of clusters so the one-to-one correspondence between chain and dominant variants is lost. In other words We are no longer sure that a new chain will define a dominant variant.

In Table 3.5 the best values for the threshold,the cut, the total number of clusters and the number of candidate alpha variant chain has been reported. We define the level of confidence as the inverse of the number of candidate. This number represent the probability of a new chain to be a new dominant variant as shown in Figure 3.24.

Using a weekly approach we were able to identify the alpha variant few ($\simeq$ 4) weeks earlier with respect to monthly approach. This reduction comes mainly from the weekly binning choice, further time can be recovered but with a strong reduction in the confidence level.

Week	Threshold	Cut	N. of Clusters	N. of candidates
42	20	0.0005	12	9
43	60	0.03	5	2
44	100	0.01	2	1

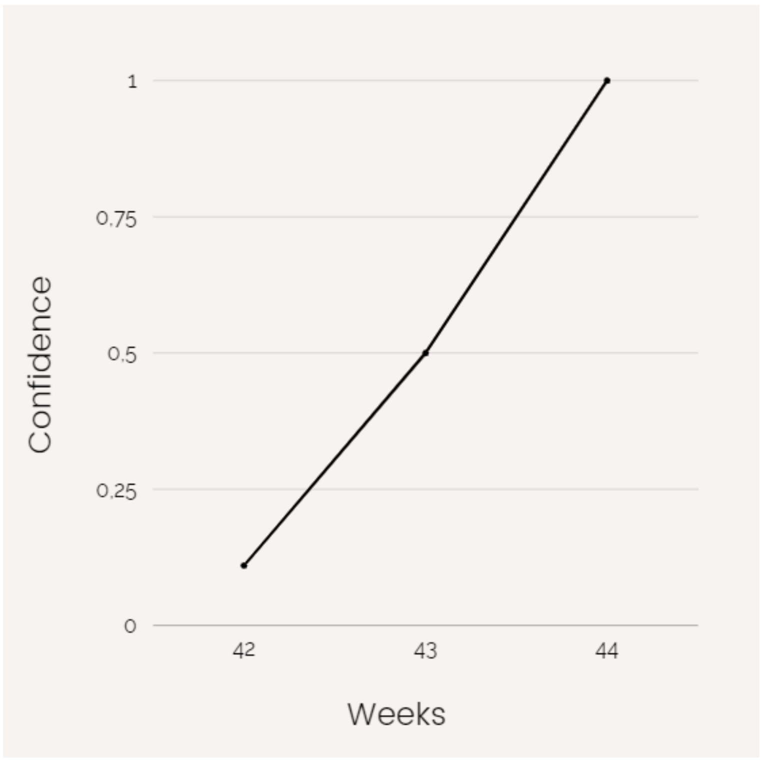


Figure 3.24: Level of confidence in percentage per week

In principle it is possible to perform the same analysis also on other dominant variants as the v1a and v1b but this time the available dataset size is considerably reduced and we are affected by large statistical fluctuations that prevent us to give a reasonable estimation of the confidence level for v1a and v1b.

3.6 Biological Studies

For each time-ordered chain it has been possible to investigate the local path of mutations along the time. This way we investigated the historical trends that leads to the origin of new variants.
We can visualize the mutations hot-spots (the position in which the sequence varies the most) and see which variants are associated to which mutations. In particular the heat-maps of mutations for the v0 and v2 have been studied.

We need to find a way to normalize our data, since as show in the previous section, the number of sequences increases and decreases during the time. In particular, the sequences collected for the alpha variant (v2) are more numerous than the ones collected for v0.

To do this we applied the following procedure:
We define a vector counter (dimension = 1273, the number of the sites in the spike protein) and initialized it to 0. Our goal is to increase this counter in the sites in which a mutation occur. To do so:

1. For each cluster i select the sequences repeated more than 1% of the total elements in the cluster
2. Select the sequences repeated more than 1% in cluster $i - 1$
3. For each sequence selected in cluster i look for its more similar correspondence in cluster $i - 1$, such that the Levensthein distance between these 2 sequences is the minimum: now we have Seq1 associated to cluster i and Seq2 associated to cluster $i - 1$
4. Scan together these 2 sequences if we find a mismatch of amino acid in the same position we increment the counter in that position taking into account the number of sequence $Seq1$
5. This procedure is repeated for all the sequences selected in cluster i
6. At the end we divide the counter by the number of all elements present in the cluster
7. We had to saturated our results for visualisation purpose, so every values above 0.1 are lowered to this value

This way we have defined a probability of mismatch for each of the 1273 sites of the sequence. The sequences repeated more than 1% of the total number of elements in the cluster are denominated sub-dominant variants.
Instead, for the first cluster in each chain we used as reference the most common sequence in the the parenthood cluster defined using branching links in sec.3.4.4 while for v0, we used as reference the sequence from Wuhan to compute the mismatch for the first cluster.

In Figure 3.25 is shown the mutations heat map for the chain associated to v0. It is interesting to notice the mutations that occur at the beginning of the sequence which represents the Peptide signal. It is possible that during the evolution of the time ordered chain through time mutations appear and disappear in the same site. In this sense, it is like the virus is sampling mutations site randomly but does not

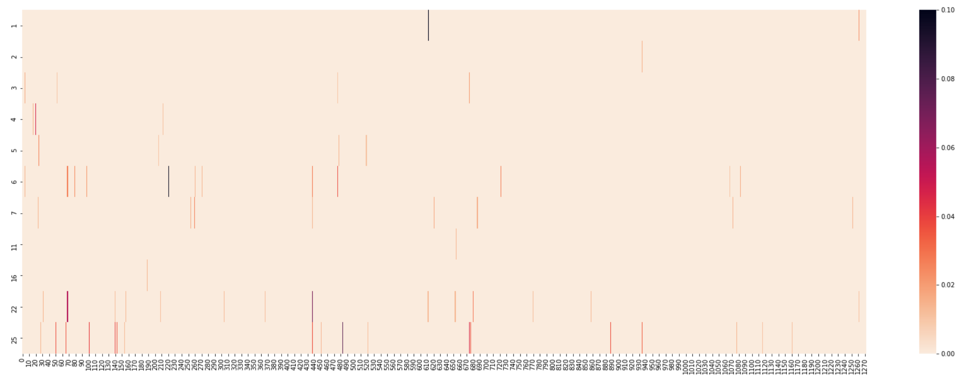


Figure 3.25: v0 Hotspots

improve very much the transmission rate.
Also, in cluster 6 we can see a darker hot-spots at the site 69-70 that are the same sites interested in the mechanism that leads to the v2 variant through an intermediate step as shown in fig.3.13. Besides, we can notice that sites 69-70 reach again a bigger value of mutations rate later in time (cluster 22) when the v2 chain was already defined and well separated from v0 chain.

Moreover, we can visualize the same heat-map for the v2 (alpha variant). Also here, although the protein results more stable wrt vo chain we can recognize some sites where recurrent mutations appears.

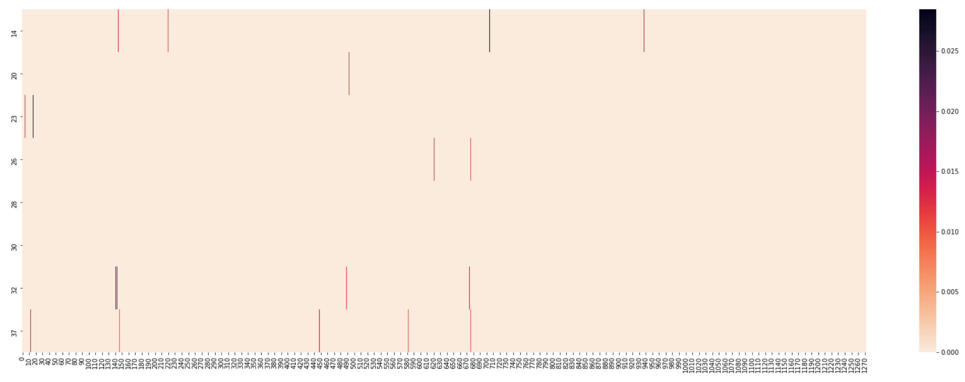


Figure 3.26: v2 Hotspots

Chapter 4

Conclusions

The aim of this project is to use a machine learning approach to study the time evolution and the dynamics of Covid-19 pathway finalized to predict future further mutations. In particular in this analysis the spike protein has been taken into account since it plays a key role in the virus interaction with the human body [39].

To predict the time evolution of the spike protein and to identify possible dangerous mutations we need to reconstruct the evolutionary path of the sequences. For this reason, firstly a clustering procedure has been applied to the whole dataset to construct time series sequences and then a Machine Learning algorithm will be trained using such time-series as input.

Looking deeply at the reconstructed time-series it has been also possible to improve our understanding of the evolutionary path of Covid-19 variants, together with how and when they born and die. Moreover, it is noticeable to consider how they interact with each other and how they generate from each other. With further analysis it is possible to have a nice picture of the competitive behaviours of viruses, and in this way to show that in fact as a virus mutates it can be identified as a new virus indeed. In conclusion, this approach is a robust way to study the behaviour and time evolution of different families of variants.

Starting from the Gisaid proteomics data, we first divided them into monthly time slots and then performed hierarchical clustering on the amino acid sequences, then we built connections between clusters in consecutive months by defining a new algorithm to link them together in unambiguous way. However the granularity of our analysis can be changed, as an example a weekly based analysis has been performed in order to have faster and more accurate tool to predict the spread of a variant of concern.

The strength of this approach relies on the possibility to build time evolution of candidate variants without any prior bias. So, we firstly tune the clustering algorithm finding the best working points using only stability criteria and then we were automatically able to find a correspondence between our dominant variants and some variant of concern.

Using this approach it has been possible to recover time information of a candidate variant, and for each candidate variant it is also possible to define a parent variant to better track the evolutionary path. This way the proposed method allow to identify some variants of concern and interest together with their evolutionary path and it represent a first attempt towards the development of more complex strategies to study the evolution of variants in the current pandemic.
In this work we used only the amino acid sequence of the spike protein, so the computational method is less intensive rather than standard algorithms that investigate the full nucleotide sequences of RNA. Hence the proposed method represent a light tool that can be used to study the time evolution of different variants, without having any biological background about the structure of the protein.

Studying the evolution of the genesis and the deceasing of the variant we can investigate the nature of the ongoing pandemic and see the forces that drive each pandemic wave, in fact with the right threshold we can see that each pandemic wave is guided by a new emerging variant. In this sense we can speak of a real mutation driven evolutionary model. In which a variant emerges, overcome the previous ones and then proliferates until a new variant emerges and so on.

To summarize with this method we were able to track the evolutionary path and the parenthood of different variants, moreover results have been validated taking into account more data from different region of the world and finding a very good correlation with the nomenclature released by WHO and Gisaid. Indeed the proposed method allowed us to successfully identify the evolutionary pathway of some variants of concern (like the alpha and the delta VoC) and interest, together with other variants that were not identify with the standard methods. At the end, we are able to decompose each pandemic wave into sub waves,and see the contribution of each variant to the total number of infected. Moreover the approach achieves very good results compared to variants classification from WHO [40] and GISAID. Thus this simple machine learning algorithm seems good enough to identify variants of concern and interest, and see how relevant variants emerge from the older ones and construct phylogenic relation among variants.

Moreover, this approach has been validated using Covid-19 data but it is not specific for this kind of virus. In this sense, having enough data for other infections (like the influenza viruses) we might be able to translate this approach to other disease, and see the spread and the genesis also for their variants.

Moreover, biological interpretation of these chains and variants are proposed in terms of hotspots, looking at the regions of the proteins that are less subjected to variations. In this terms, it can be used to improve the efficiency of the vaccination strategy, by creating vaccines that targets these regions .

At the end, we proved that with a weekly based analysis we are able to identify the arising of a new VoC with advance respect to the standard procedures. In fact, we are able to correctly identify the alpha variant in November 2020, instead it has been named a VoC only in December 2020.

The machine learning analysis allows us to naturally integrate the time evolution of virus variants and their genesis into the eRG framework discussed at the beginning. This leads to a coherent picture of how temporal symmetries are key to understand not only the overall epidemiological understanding of a pandemic but also its atomic version in terms of the virus variants. In other words we have now an epidemiological theory of variants based on fundamental physics principles.
Indeed this work represent a first attempt towards of an exploratory strategy of the genesis of variants.

4.1 Future developments

As discussed in the previous chapter in this analysis we used the UK dataset because only in that case it has been possible to retrieve a good amount of data, i.e. enough to keep the statistical fluctuations related to the procedure under control. In some sense this represent the main limitation of this work and having a larger amount of data from other regions to process could be very relevant to further validate the proposed method and will lead to further improvement.

Using this work as template it is possible to deepen the analysis looking not only at the aminoacid sequence but also at the 3d structure itself.
In fact, a spike protein is formed by three aminoacid chains, that define its properties and functions. Substituting one amino acid on the head of this architecture, for example where the BPD domain is present may lead to a more significant change rather than a mutation that appears in the stem.
For this reason, it will be useful to look at the time evolution of the structure, taking into account which kind and where the modifications will occur for each variant of interest.

Another possible future work (provide that data will be accessible) rely on the application of this approach to other biological systems.
In particular, we would like to see if cancer dynamics is ruled by the same eRG approach. In fact, cancer is due to cells that start to grow abnormally and fast reproduce themselves, and this leads to a very high mutation rate.

4.2 A Variants prediction tool

This part is the backbone of an ongoing project that aims to create a neural network approach to predict future mutations.
The idea is to use the reconstructed time series from the epidemiological data described in chapter 3 to feed a neural network in order to understand the relevant patterns of mutations in the spike protein structure that can be related with new dangerous variants. In this sense the project aims to building a new method to early understand the genesis and the spreading of viruses.

Once we have the temporal pathway, we can exploit all this information to study the virus evolution and predict some dangerous mutation.

There are a lot of useful patterns we can extract from our dataset indeed the first part of this approach is devolved to understand which kind of pattern is more relevant for our purposes:

- We can investigate the hidden pattern of the virus variants, and hope to find some rules that govern its behaviour.
- We can perform a local analysis on a given region of the protein, in fact we know which sites are varied the most or have leaded to a dangerous mutation.

And this is just two of the biological questions we can ask ourselves.

4.2.1 Data preprocessing and Embedding using ProtVec

In order to properly feed the neural network the reconstructed time series of sequences need to be transformed from string of chars to numeric representation. To do so we exploit the work of ProtVec [41].
ProtVec is a Continuous Distributed Representation of Biological Sequences that exploits word2vec representation.
Following the work of [42], we start from the aligned sequences (this way all the sequences have the same length) and divide each of them in overlapping 3-grams.

Here for example from the sequence:

'MFVFLVLL'

we retrieve the following 3-grams:

MFV FVF VFL FLV LVL VLL

In this sense, we can see each sequence as a phrase composed by overlapping 3-grams.
Given the 20 common amino acids that are encoded by the codons the dictionary width (i.e. all the possible different possible 3-grams) is equal to 20^3. Each sequence (length = 1273) could take a maximum dictionary occupancy of about 5%. All the possible different 3-grams we found in our dataset will be used as corpus to fed the Word2Vec algorithm.

The Word2Vec algorithm has been trained using a windows for the context words with a size of 3 elements and setting the dimension of the hidden layer to 100. After this step, each 3-gram is represented by an array of 100 floating numbers. The dictionary is built using all the sequences under analysis. This means that if we are analyzing data until month 10 we need to use as corpus all the sequences until this month.

To summarize: After the embedding each sequence is represented by the list of all its ordered 3-grams and it is possible to unequivocally identify a sequence in a numerical representation in the form:

$$\text{sequence} \rightarrow [1273 \otimes 100]$$

4.2.2 Visualization

The proposed embedding method allow us to represent each sequence as the summation of the individual 3-grams thus with a vector of dimension equal to 100. To be sure we are feeding the neural network with an embedding that is coherent with our previous clusterization we used a visualization tool.
As already said we define a sequence as the sum over all its 3-grams, this way we associate each sequence to a 100 elements array.

$$\text{sequence} \rightarrow 100$$

The dataset has bee normalized and then a PCA has been applied to reduce the dimensionality of the matrix to 50. This step has been used to speed up the

computional time but it is also needed to avoid the curse of dimensionality on the Euclidean distance used in t-SNE, that may lead to not be able to divide correctly the points.
Finally we performed a t-SNE algorithm to project the dataset on a simple two dimensional space this way we are able to plot the sequences on a plane.
Since t-SNE is a stochastic algorithm, it was chosen to remove repeated sequences from the dataset to avoid the unwanted effect of getting the same sequences projected in different positions of the two dimensional plane. The main reason to use the t-SNE approach is to to preserve the local neighborhood of our dataset.
In Figure 4.1 we can see the results of t-SNE algorithm applied for month 10. Each point represent a sequence in this new projection while the color represent the ouput of the clusterization procedure described in chapter 3. The nice compatibility between the clusterization method and the ouput from the embedding procedure is quite evident. Sequence belongig to diffrent clusters are well separated also in the t-SNE representation.

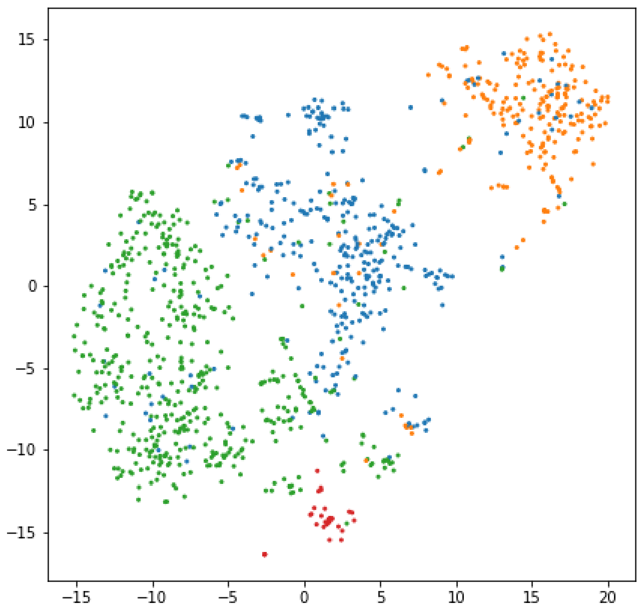


Figure 4.1: t-SNE results for all sequences in month 10.The color represent the ouput of the clusterization procedure described in chapter 3

4.2.3 LSTM implementation

Now that we passed from strings of different lengths into an equal numerical representation, the data are ready to feed the Recursive Neural Network (RNN) [43]. We proposed RNN with LSTM which is a deep learning algorithm that is able to manage the long terms dependencies while preserving the short-term information about the data in input and to address the complex dependencies of the time series.

As usual to train the model, the dataset is divided into 3 subgroups called train, validation and test set. The sequences from the train set are extracted randomly from consecutive clusters belonging to the same chain as shown in Figure 4.2. This way each sample is represented by a time serie of sequences $X_1....X_n$ and it is given as input to the RNN algorithm which is trained to minimize the loss on the validation set i.e. to minimize the mismatch between predictions and ground truth. We would like to use the RNN algorithm to give an estimation of the probability of mutation at site **k** and time **t** given the time series $X_1^k....X_{t-1}^k$. As a first attempt we decided to focus our attention on the second question, so we would like to use RNN to learn the hidden pattern of mutations for a small region of the spike protein.

So the algorithm should learn this pattern from a list of sequences as input and, using then, then predict the probability of a given amino acid in position i.
I used the RNN architecture that is typical of natural language process since the problem of understanding pattern of amino acids can be seen as similar to understanding the meaning/semantic of words into a phrase and to early identify or predict change in the spike protein structure that can be interpreted as a semantic change.

The algorithm works in the following way, supposing that we want to predict the behaviour of an amino acid in position i in month k, we utilize the information about the k-1 month before:

- Input: a list of k-1 sequences, from each cluster we randomly sample one sequence and concatenate them together
- Label: select random a sequence in cluster m, looking at the element in position i
- Output: probability distribution

This part is still a work in progress, so I am not yet able to show any results.

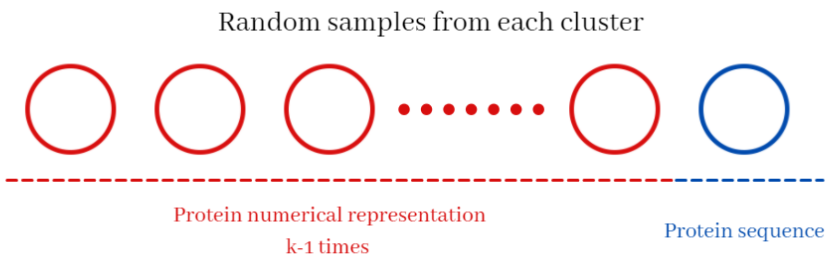

Random samples from each cluster
Protein numerical representation
k-1 times
Protein sequence

Figure 4.2: Dataset creation

Chapter 5

Code

5.1 Repository

The code is available at https://github.com/AdeledeHoffer/ML-Covid

Bibliography

[1] *RNA regions figure.* `https://www.ncbi.nlm.nih.gov/books/NBK554776/figure/article-52171.image.f5/` (cit. on p. 2).

[2] *Covid 3D figure.* `https://www.unisr.it/news/2020/3/viaggio-al-centro-del-virus-come-e-fatto-sars-cov-2` (cit. on p. 3).

[3] Huang Yuan et al. *Structural and functional properties of SARS-CoV-2 spike protein: potential antivirus drug development for COVID-19.* 2020 (cit. on pp. 3, 4).

[4] Pokhrel S., Kraemer B.R., and Burkholz S. et al. *Natural variants in SARS-CoV-2 Spike protein pinpoint structural and functional hotspots with implications for prophylaxis and therapeutic strategies.* 2021 (cit. on p. 3).

[5] Yuan Huang, Chan Yang, Xin-feng Xu, Wei Xu, and Shu-wen Liu. «Structural and functional properties of SARS-CoV-2 spike protein: potential antivirus drug development for COVID-19». In: *Acta Pharmacologica Sinica* 41.9 (2020), pp. 1141–1149 (cit. on p. 4).

[6] Rafael Sanjuán, Miguel R Nebot, Nicola Chirico, Louis M Mansky, and Robert Belshaw. «Viral mutation rates». In: *Journal of virology* 84.19 (2010), pp. 9733–9748 (cit. on p. 4).

[7] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. «Basic local alignment search tool». In: *Journal of molecular biology* 215.3 (1990), pp. 403–410 (cit. on p. 5).

[8] Jeffery K Taubenberger and David M Morens. «1918 Influenza: the mother of all pandemics». In: *Revista Biomedica* 17.1 (2006), pp. 69–79 (cit. on p. 5).

[9] Giacomo Cacciapaglia, Corentin Cot, Michele Della Morte, Stefan Hohenegger, Francesco Sannino, and Shahram Vatani. *The field theoretical ABC of epidemic dynamics.* 2021. arXiv: `2101.11399 [q-bio.PE]` (cit. on pp. 5, 11, 15).

[10] William Ogilvy Kermack and Anderson G McKendrick. «A contribution to the mathematical theory of epidemics». In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721 (cit. on p. 5).

[11] *SIR figure.* `https://www.researchgate.net/figure/SIR-model-Schematic-representation-differential-equations-and-plot-for-the-basic-SIR_fig2_47676805` (cit. on p. 8).

[12] G. Cacciapaglia, C. Cot, and F. Sannino. «Second wave COVID-19 pandemics in Europe: a temporal playbook». In: *Scientific Reports* 10 (2021) (cit. on pp. 7, 10).

[13] G. Cacciapaglia, C. Cot, A. de Hoffer, S. Hohenegger, F. Sannino, and S. Vatani. «Epidemiological theory of virus variants». In: *Arxiv* (2021) (cit. on pp. 8, 13, 14).

[14] Michele Della Morte and Francesco Sannino. «Renormalisation Group approach to pandemics as a time-dependent SIR model». In: *arXiv preprint arXiv:2007.11296* (2020) (cit. on p. 10).

[15] Giacomo Cacciapaglia and Francesco Sannino. «Evidence for complex fixed points in pandemic data». In: *Frontiers in Applied Mathematics and Statistics* 7 (2021), p. 42 (cit. on p. 10).

[16] Michele Della Morte, Domenico Orlando, and Francesco Sannino. «Renormalization group approach to pandemics: The COVID-19 case». In: *Frontiers in physics* 8 (2020), p. 144 (cit. on p. 10).

[17] Sandeep Hosangadi. «Distance Measures for Sequences». In: *arXiv* (2012) (cit. on p. 16).

[18] *BioPython.* `https://biopython.org/docs/1.75/api/Bio.pairwise2.html` (cit. on p. 17).

[19] *Pairwise alignment figure.* `https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f/` (cit. on p. 18).

[20] Anna Szymkowiak, Jan Larsen, and Lars Kai Hansen. «Hierarchical clustering for datamining». In: *Proceedings of KES-2001 Fifth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies.* 2001, pp. 261–265 (cit. on p. 18).

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013 (cit. on pp. 18, 20).

[22] *Hierarchical clustering figure.* `https://www.displayr.com/what-is-hierarchical-clustering` (cit. on p. 19).

[23] *Dendrogram figure.* `https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/` (cit. on p. 20).

[24] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning.* The MIT Press, 2012. ISBN: 026201825X (cit. on p. 20).

[25] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012. ISBN: 0262018020 (cit. on pp. 20, 22).

[26] Laurens Van der Maaten and Geoffrey Hinton. «Visualizing data using t-SNE.» In: *Journal of machine learning research* 9.11 (2008) (cit. on p. 21).

[27] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* `http://www.deeplearningbook.org`. Cambridge, MA, USA: MIT Press, 2016 (cit. on pp. 22, 23, 25, 26).

[28] *Perceptron figure.* `https://datascience.eu/it/apprendimento-automatico/perceptron/` (cit. on p. 23).

[29] *ANN figure.* `https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051` (cit. on p. 24).

[30] *RNN figure.* `https://www.researchgate.net/figure/Schematic-diagram-of-a-simple-RNN-network_fig1_283761596` (cit. on p. 27).

[31] *RNN figure.* `https://www.easy-tensorflow.com/tf-tutorials/recurrent-neural-networks/many-to-one-with-variable-sequence-length` (cit. on p. 28).

[32] *RNN figure.* `https://kharshit.github.io/blog/2019/01/04/the-gradient-problem-in-rnn` (cit. on p. 28).

[33] *LSTM figure.* `https://www.researchgate.net/figure/Structure-of-the-LSTM-cell-and-equations-that-describe-the-gates-of-an-LSTM-cell_fig5_329362532` (cit. on p. 29).

[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space.* 2013. arXiv: `1301.3781 [cs.CL]` (cit. on p. 29).

[35] *Word2Vecfigure.* `http://naviglinlp.blogspot.com/2021/03/lecture-6-11032021-2-hours-word2vec.html` (cit. on p. 30).

[36] *GISAID.* `https://www.gisaid.org/` (cit. on p. 31).

[37] *Pangolin Lineage.* `https://cov-lineages.org/` (cit. on p. 44).

[38] *CDC Variants.* `https://www.cdc.gov/coronavirus/2019-ncov/variants/variant-info.html` (cit. on p. 44).

[39] Adele de Hoffer, Shahram Vatani, Corentin Cot, Giacomo Cacciapaglia, Francesco Conventi, Antonio Giannini, Stefan Hohenegger, and Francesco Sannino. *Variant-driven multi-wave pattern of COVID-19 via Machine Learning clustering of spike protein mutations.* 2021. arXiv: `2107.10115 [q-bio.GN]` (cit. on p. 61).

[40] *WHO*. `https://www.who.int/` (cit. on p. 62).

[41] Ehsaneddin Asgari and Mohammad R.K. Mofrad. «ProtVec: A Continuous Distributed Representation of Biological Sequences». In: *PLos ONE* (2010) (cit. on p. 64).

[42] Rui Yin, Emil Luusua, Jan Dabrowskiand Yu Zhang, and Chee Keong Kwoh. «Tempel: time-series mutation prediction of influenza A viruses via attention-based recurrent neural networks». In: *Bioinformatics* (2020) (cit. on p. 64).

[43] Sepp Hochreiter and Jurgen Schmidhuber. «Long short-term memory». In: *Neural Computation* 9 (1997), pp. 1735–1780 (cit. on p. 67).