Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Synthetic data generation for the assessment of antimicrobial resistance through machine learning

Supervisor:
Dr. Claudia Sala

Submitted by:
Adriano Zaghi

Co-supervisor:
Prof. Gastone Castellani

Academic Year 2021/2022

Contents

Abstract

Antibiotics are a resource that has been revolutionary for medicine and biology since those allows to control bacterial infections on humans and in all kinds of environments. Apart from medicine, antibiotics are exploited also in production techniques, like farming, agriculture and food manufacture. Despite the wide use of them, it is well proven that popular antibiotics are becoming less effective on their targets; this seems to be a consequence of the exposure of the bacteria to those substances. Due to an (un)natural process of species selection, bacteria are gaining resistance to the more diffuse antibiotics, at the point that, in many situations, we cannot afford to distribute new antibiotics, in order to preserve their effectiveness. AntiMicrobial Resistance (AMR) is becoming a serious threat for human and in general for the ecosystem, being responsible for 1.27 million deaths in 2019 [1]. To monitor and predict AMR is hence a fundamental part in the effort of avoiding pandemic situations which have proven to be problematic in a globalized society: metagenomics analysis seems to be the way to do so. As a consequence of the diffusion of next generation sequencing techniques, metagenomics databases have become indeed one of the most promising repositories of information about features and behavior of microorganisms. One of the subjects that can be studied from those data are bacteria populations. Next generation sequencing techniques allow to study the bacteria population within an environment by sampling genetic material directly from it, without the needing of culturing a similar population in vitro and observing its behavior. As a drawback, it is quite complex to extract information from those data and usually there is more than one way to do that; AMR monitoring is no exception. In this study we will discuss how the quantified AMR, which regards the genotype of the bacteria, can be related to the bacteria phenotype and its actual level of resistance against the specific substance. In order to have a quantitative information about bacteria genotype, we will evaluate the *resistance spectrum* of bacteria from the read libraries, aligning them against CARD database. With those data, we will test various machine learning algorithms for predicting the bacteria phenotype. The samples that we exploit should resemble those that could be obtained from a natural context, but are actually produced by a read libraries simulation tool. In this way we are able to design the populations with bacteria of known genotype, so that we can relay on a secure ground truth for training and testing our algorithms.

Chapter 1

Introduction

DNA was isolated for the first time in 1869 by Friedrich Miescher, but only in the middle of 1900 scientists started to figure out its importance in the constitution of life. Two milestones in this root where placed by Oswald Avery, Colin MacLeod, and Maclyn McCarty[2], who noticed that purified DNA could change a strain of bacteria into an other, and James Watson and Francis Crick[3], who discovered the structure of this protein. While the second publication is well know to a large audience, also the first can be considered as a fundamental starting point for the exploration of the field we are approaching in this work. During the last 70 years the technologies improvement allowed a deeper and deeper insight on the structure of DNA and on how it determines the behavior of cells. It is worth to mention some achievements in this field:

- Sequencing of the first genetic sequence, associated to insulin (by Frederick Sanger in 1955) [4]
- Sequencing of the first complete genome from a DNA based virus (again by Frederick Sanger in 1977) [5]
- Sequencing of the first complete genome of a free-living organism, the bacterium Haemophilus influenzae (1995 by Hamilton Smith et all) [6]
- Sequencing of the first complete human genome in 2003 (by Human Genome Project)
- Diffusion of high troughtput sequencing procedures: next generation sequencing

With "sequencing" we are referring to the practice of detecting the sequence of bases that are enclosed in the double helic of the DNA. This indeed is the most relevant part of the structure proposed by Watson and Crick and we are convinced that the properties of the DNA can be deduced from those. The result of a sequencing procedure is essentially a sequence of symbols that represents the four different bases of the DNA (A, T, C, G);

in some situations, a quality score is associated to the information of each base. We will not get into the details of how the sequencing procedure is performed, but is worth mentioning that, at the moment, there is an reasonable availability of those data, from the most various sources. When the genetic material of populations of cell is sequenced we obtain big samples that encode genetic sequences: we call them genomic data.

In this study we will focus on a particular kind of genomic data: those obtained from bacterial populations. Bacteria are relatively simple organisms, composed by just one cell, and are very important in a wide variety of situations. They are present in almost every place of the biosphere and are fundamental in various processes that guarantee the persistence of the ecosystem; moreover almost all the multicellular organisms lives in symbiosis with bacterial populations located in various places of their body (skin, stomach, gut. . .). On the other hand, bacteria populations which are harmful for human or other organisms can become a serious threat for them. The reason is their explosive reproduction rate in appropriate environmental conditions and their adaptability. This problem becomes even more important in a globalized society, in which humans and goods travels around the world every day, spreading and mixing the bacteria that they carry with them.

Such an important component of the biosphere have to be controlled and regulated in order to avoid pandemic situations and the most effective tool that we have elaborated are the antibiotics. With this generic term, we refer to antimicrobial substances which are active against bacteria and are able to eliminate them, or at least reduce their reproduction rate. Of course, as there is a variety of bacteria, there is also a variety of antibiotics and the effect of them depends on their target. The research of more efficient antibiotic substances is a primary importance field in medicine and is probably the most direct attempt to regulate bacteria activities. Unfortunately, using strong antibiotics will not be enough to control bacteria populations, due to the fact that bacteria have an incredible capability of adapting and evolving. It is indeed well proven that the effectiveness of popular antibiotics have decreased in time and this is surely connected to the exposition of bacteria populations to them. The high reproduction rate of bacteria allows their community to adapt to harmful environmental conditions, including antibiotics substances. This adaptation takes place because in each reproduction cycle many bacteria are affected by mutations and some bacteria can even mutate during their life exchanging genetic information with others. While many of those mutations are fatal or useless to the individual, some of them can give him an advantage against an antibiotic substance that used to be effective. The intense use of antibiotic then produce the effect of selecting those resistant bacteria eliminating the others; thus the antibiotics that used to be effective are now very less powerful on those selected strains. This phenomena is called antibiotic resistance and it falls in to the bigger category of AntiMicrobal Resistance (AMR) which is the core of this study. To have a idea about how AMR is problematic at the moment, we can consider a report from the U.S. association Centers for Disease Control and Prevention (CDC). The analysis shares that more than 2.8 million antibiotic-resistant infections occur in the U.S. annually, and 35,000 people die as a result. Six multidrug-resistant germs alone have had an immense cost for the U.S. health system: more than $4.6 billion every year. [7].

As proven by the experiment performed by Avery, MacLeod and McCarty and many other evidences, the behavior of bacteria is determined, in relevant part, by their genetic material. This is why in various study the data provided by DNA sequencing techniques are chosen as indicator to monitor and to predict AMR. Usually the starting point for the assessment of AMR in a wild bacteria population are the data produced by the next generation sequencing techniques; those are one of the high troughtput technologies mentioned before. In order to read huge amount of bases in a certain time, next generation techniques sequence the DNA in parallel, reading multiple sequences at the time. The drawback is that those techniques are capable of reading only short fragments of DNA, which we call reads. The final output of those procedures are big read libraries that contains the short sequences (between 100 and 200 bases) and the information about their quality. The total size of those read libraries is calculated in GigaBasePair (1 Gpb = 10^9 bases encoded); in this study we used 1 Gbp read libraries, but in many cases those are even bigger. Notice that in this procedure we can not keep track of the organism that owned the genetic material from which the read is sampled. This aspect of the data is very important, since we have no possibility of linking the data we are dealing with to a specific organism without taking into account other databases and supplemental information. The data produced by shotgun sequencing shall be instead considered to be about a population of bacteria. In many situations, the fact that we can not assign the genetic material to a specific source will be problematic, but we should say, anyway, that a single bacteria, even with very high resistance, has very little relevance: almost all the parameters that we can be interested in should then be referred to the population or to a part of it.

In this study we will mention two kind of bacteria population which are completely different and shall not be confused: the in-vitro populations and the wild populations. First kind of bacteria populations are cultured in laboratory and are composed by a large amount of bacteria which are almost equal to each other: those can be considered clones. Those populations are useful for various kind of experiments and in particular to test the effect of antibiotic substances. After the tests the genome of the bacteria, which should be almost equal for all the clones, is sequenced and creates a reference for that specific population. For a single bacteria specie, a variety of different populations can be cultured, all with relevant differences: we call them strains and usually the genomes that we find in the databases refers to a particular strain.

Our study will instead be centered on the wild populations of bacteria. Those are characterized by a great heterogeneity of species and strains that have diverse abundance and diverse behavior. All together determine the behavior of what we call micro-biome. Depending on the circumstance, it can be useful to refer the AMR to the whole population, to a single specie that compose it or to a strain of a certain specie. It is obvious that, dealing with a population of different individuals, rather then a uniform one (in-vitro populations) increases the complexity of the metagenomics data and makes more difficult to obtain useful information from them. Anyway, the sequencing procedure performed in order to obtain metagenomic data from wild populations is essentially the same procedure used for in silico population DNA sequencing and do not present additional difficulties.

At the moment, a huge amount of metagenomic data, both from in-vitro and wild population is available. The data from wild populations have been extracted from the bacteria found in a variety of environments, like human or animal gut, marine or sewer water and also food. Evaluating the information about the AMR from those data would give us an important insight about the effect of the antibiotics that we have used up to now and also an advice about what antibiotics to use in the future. Never the less, the objective of this study is not to access the resistance of a specific population, but we are interested instead in evaluating the performances of different AMR quantification techniques and this is why we will work with simulated data. Part of the study has been indeed to implement a tool capable of simulating the output of a shotgun sequencing procedure on a population designed from us. This allowed us not only to compare the different AMR evaluation algorithms between them, but we could also validate them against our ground truth about the simulated data.

Chapter 2

Materials and methods

In the following sections, we are going to present the tools that were used to perform the analysis; those will be explained following the order in which are applied on the data. This organization reflects the structure of the computational pipeline that have been implemented in order to perform the analysis. The software Snakemake is probably the best way to deal with the complex sequence of tasks required, allowing to split them in different modules (rules). Snakemake provides a lot of facilitation and useful features, like paralleling of multiple analysis on various data and ad hoc environment construction. Anyway, the most relevant aspect is the fact that the Snakefile, which is the file where the pipeline can must be encoded, is essentially a track of the operations performed on the data. This and the information about the software distributions that was used, makes the pipeline reproducible and allows in any moment to perform again the same analysis. Being reproducible is fundamental in scientific experiments and the metagenomics ones are no exception. Here [8](`https://github.com/AdrianoZaghi/AMR`) can be found a link to a github pace where is stored the Snakemake file encoding the pipeline we used and all the other environment files and documentation required to make it work. We provide also an image 2.1 of the

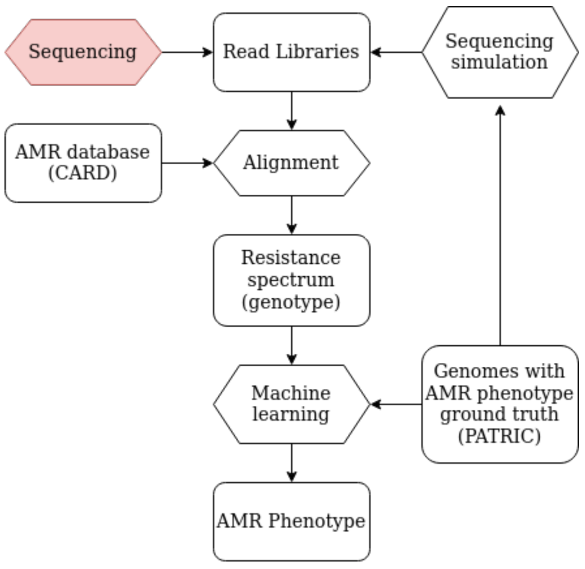


Figure 2.1: We present a flow chart of the metagenomic data processing. Rectangular boxes represent data while the hexagonal boxes represent procedures. The "Sequencing" box is in red, since we did not performed that procedure, but it should be the sarting point of the pipeline once it is used on real world data

flow of the data in the piepeline, so that the reader can have a better idea of it.

2.1 Genomes information

In this study we intend to identify a sequence of algorithms capable of predicting the AMR phenotype (resistant or not to a specific substance) of a bacteria population starting from the reads that we obtain from a shotgun sampling of its genetic material. The starting point to do so is to find a database containing the same kind on information that that we intend to predict connected to our starting data; in this case the AMR phenotype of a bacteria, associated with the genome sequences of that. Our choice was PATRIC database [9] (PathoSystems Resource Integration Center). Funded by the National Institute of Allergy and Infectious Diseases, PATRIC gives access to over 250 000 uniformly annotated and publicly available genomes with curated metadata. Between all the information about a genomic sequences, we have been interested in the AMR phenotype relative to the *Imipenem* antibiotic, a member of the carbapenem antibiotic family. The choice of this specific substance was due to the availability of many genomess having known AMR phenotype about it. More specifically, we focused our attention on different strains of the *Acinetobacter Baumannii* (NCBI ID 470). This bacteria has come to the attention to the scientific community because it proliferates in hospitals [10]. Its resistance mechanisms are widely studied and we found indeed a lot of data about different strains of it. The strains that was used in this study had the AMR phenotype information relative to the *Imipenem* noted in their metadata [11].

This information is evaluated by laboratory studies or exploiting machine learning classifiers. In PATRIC can be found a lot of genomes annotated with the AdaBoost Classifier, but our choice was not to consider them. Indeed it seems redundant to train a machine learning algorithm on data produced by an other algorithm of the same kind. We are left then with the genomes whose phenotype was evaluated by physicians in laboratory. This procedure is relays on the culturing of the other kind of population we mentioned before, the clones population. The effects of an antibiotic substance is tested exposing those populations of very similar individuals to it. There are two techniques that are used in this process and those produces different results.

- **Disc diffusion**

  Bacteria populations are cultured on a solid soil. With successive selections, the population is purified up to the required homogeneity. On the solid support are then placed fixed quantities of the substances of which we intend to test the resistance of the population; those are contained in small disks of tissue. If the bacteria are susceptible to the substance, those which are near the tissue disk will die and a circular surface around the disk will depopulate. The more the bacteria is susceptible to the substance, the bigger will be the diameter of this surface. The result of this procedure is indeed the measure of this diameter.

- **MIC**

  Minimum Inhibitory Concentration is the result of a different kind of experiment that aims to measure AMR in bacteria. Again, the bacteria population is selected in order to achieve a certain level of homogeneity, but in this case the culture is on a liquid soil, weather then a solid one. If left unchecked, the bacteria population will reproduce and increase it's number of individuals, since the culture is, by definition, a favorable environment. To access the AMR phenotype of the cultured bacteria to a certain substance, the antibiotic is added to the liquid culture. The MIC is the minimum concentration of the substance required for stopping the increase of bacteria number.

As it can be noticed, the results of those two measurements is quite difficult to compare, if not by making both experiment on the same population, and this is rarely the case. Moreover, in the present study we did not consider a continuum quantity as a measure of AMR phenotype, but rather a discrete variable: "susceptible" or "resistant". This information is evaluated by the two type of data mentioned before by establishing thresholds. The categorical definition of AMR phenotype is the most diffuse, since it can be obtained by both kind of procedures. It is also the one that is usually evaluated by machine learning techniques, like the AdaBoost Classifier used by PATRIC itself.

PATRIC, like all others -omic databases, collects an impressive amount of evidence from biomedical studies. Due to such high numbers and variety, it results quite difficult to ascertain if all the studies have followed the same procedure and chosen the same threshold. The standardization of this kind of information is the major challenge for any kind of -omic database and there is still work to do in this sense. Natural language processing, which is used to collect the articles from the whole world, and well established laboratory routine have still grate space of improvement. Never the less we trusted PATRIC information and we were able in to implement an effective pipeline capable of well classifying those data.

2.2 Community design and simulation of data

Simulating metagenomic data was the starting point of the analysis. The tool that was used to perform this task was Camisim [12], a program capable not only of simulating the reads that could have been sequenced from a given genome, but can also simulate strains of that genome. Those are actually two operations that aim both to reproduce the complexity of real world data, but are quite different to each other an thus shall be discussed separately.

The first task performed by Camisim is the strains generation. For this task, the program relies on a tool called sgEvolver [13], which is capable of reproduce in the new strains both local changes (e.g., single nucleotide substitutions and indels) and larger genomic changes (e.g., gene gain, loss, and rearrangement). Those changes to the original sequences aims to reproduce a reasonable amount of diversity inside the single specie of the population and make the data more realistic. All the mentioned phenomena are indeed types of mutations that happens frequently in a bacterial population and are at the base of bacteria evolution. Never the less, we expect them not to change significantly the AMR genotype from the original genome to the simulated strains. The simulation of strain evolution process requires as input the full genome sequences that we intend to diversify, or at least a collection of scaffolds that provides a suitable coverage. Moreover, in order to locate the provided DNA in the taxonomy classification, sgEvolver requires also the NCBI ID of the specie. This is useful in order to locate the specie in the taxonomic three provided my NCBI and exploit taxonomic considerations in order to perform a more realistic evolution. Once sgEvolver has completed its task, the strains generated are used to simulate a shotgun sampling. Notice that the original genome is not included in the simulation: this because the same population design (set of original genomes used for creating the strains and simulating a sample) will be sampled more then one time, in order to increase our sample size. By not including the original genome in the simulation, each time the sample is constructed with newly generated strain only.

Before starting the actual read simulation, we need to assign an abundance to the various strains of our sample. The number of read sampled from the genome will be proportional to such abundance. In the input data required by Camisim can be specified the abundance for a certain specie (original genome), so that we can compose the sample with different proportions of various bacteria. This abundance is then divided between all the strains generated from the original genome of that specie exploiting the so called *broken stick model.*

This model divide the abundance assigned to a genome in smaller portions, assigning each of them to a different strain generated from the genome. This approach is derived by the niche apportionment model [14] in which the resources available in a certain ecological niche are divided between different communities in a similar way as a stick can be broken in to pieces. The resources are considered to be proportional to the abundance of the population that exploits them.

In practice, when Camisim has to assign different relative abundances to the strain produced from a genome (relative to the abundance assigned to the original genome by the user), the first strain abundance is sampled from a beta distribution (Figure 2.2). The parameters of this distribution are $\alpha = 1$ and $\beta = 3$; while the first value must be 1 in order the distribution to have a shape coherent to the situation, the second value have been chosen arbitrary by us. An smaller value for β would produce very prominent abundance for the first considered strains, while a higher value would end up in a more evenly distributed scenario.

Given b_1, the first value sampled from the distribution, we will say that the first strain have a relative abundance of b_1. We are left with $1 - b$ abundance to assign to the

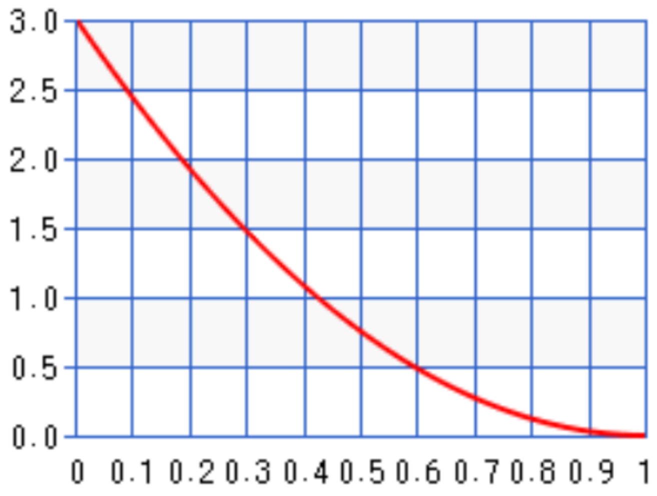


Figure 2.2: Plot of a beta distribution with parameters $\alpha = 1$ and $\beta = 3$.

remaining strains and in order to do so, we scale the "remaining part of the stick" to 1. Now the process can be repeated for the second strain, and the sampled value b_2 will be scaled back to the original measure: the second strain will have abundance $b_2 * (1 - b_1)$. If we iterate this process $N - 1$ times, where N is the number of the strains, we end up with the following partition:

$$Abundance_i = b_i * \prod_{j=1}^{i-1}(1 - b_j) \tag{2.1}$$

The relative abundance of the last strain will be set in order to reach the unit.

The generation of simulated read libraries from the genomes that we have obtained is quite a standard practice and there are many of tools that can do that. The reason of this variety is the fact that usually each tool reproduces the sampling of a specific next generation sequencing device on the market. Camisim offers various option to perform shotgun sampling simulation and we chose the one that reproduce the Illumina sequencer data: ARTillumina [15]. The reason for this choice is the fact that we have already worked with data produced from Illumina tools and those are probably the most diffused in literature. The generation of read libraries implies the sampling of fragments from the strains produced by sgEvolver: the length of the fragments is not fixed but follows a distribution with the following parameters:

- average length of 270 bases
- variance of 24 bases

The result of the simulation are reads that can not be assigned a priori to any of the genomes used in the process, since we chose the Camisim option that makes them anonymous. The only information is the one about the correspondence between the two pair end fragments, so that we were able to organize them in two pair-end libraries in .fastq format. This is how usually metagenomic data are stored and this format is always supported by any metagenomic tool.

Filtering and trimming the data was not required. ARTillumina indeed simulates just the sequences from the DNA, without any kind of adapter sequences and other artefacts that are likely to be present in a real read library and should be removed.

In conclusion, Camisim enabled us to compose various kind of bacteria population. A single population can include multiple species and for each of those species can be simulated a number of strains. For each specie, have to be submitted a genome as input to Camisim, while the strains of that specie are created by sgEvolver. From the strains are sampled reads with designed abundance and those are stored in the two .fastq files that contains the reads of a pair end library.

2.3 Fastq format and quality assessment

Read libraries are encoded in the standard .fastq format and in order to properly talk about the feature of those data is better to explain how this format is arranged. Fastq files are essentially text file and for each read the file have four lines that contains all the information evaluated from the sequencing. It is very common that read libraries are produced in pair. This means that a DNA fragment is sequenced from both ends, usually with no superposition between the forward and backward sequences. Those need to be encoded in a specific way in order to take advantage the adjacency information of the two pair end reads. Also in this study we used pair end libraries.

The items in a fastq file for a single read are organized in rows:

- First row contains the name of the name of the read and always starts with "@". The names are usually codes that creates an identity for the read inside the sample. For pair end read libraries, this identity is shared between the backward and the forward fragments and so those are supposed to have the same name. The only difference is a ".1" or ".2" at the end of the name, depending if the read have been sequenced forward or backward.

- The second raw contains a sequence of letters representing the bases which have been sequenced: "A" for adenine, "C" for citosine, "G" for guanine and "T" for timine. There might be also the letter "N", which represents a base of unknown identity. Notice that those letters might be both uppercase or lowercase. Since we are mentioning the bases letters, is a good occasion to introduce a common statistics that might be referred to the read or the the read position (i^{th} position of all reads in the library). The GC content is a proportion evaluated between the abundance of guanine and citosine nucletides. Those two bases are bound together in the double helix structure, so this proportion reflects the abundance of that couple of nucleotides over the others. The GC content is usually specific for a single specie and is used in order to identify contamination or artifacts in the read libraries.

- The third line is occupied by a "+" sign. This symbol is not informative and is the residual of an older implementation, but is kept for retro-compatibility.
- The last line contains a sequence of symbols that encodes the quality of the bases that where sequences. The fourth and the second line should always have the same number of characters. This score is related to the probability P of the sequenced base being incorrect. We will review the Illumina encoding, since this is the sequencing platform that we used, but different sequencing platforms produces different quality encoding. In our case the quality score ranges from 0 to 62 and uses ASCII 64 to 126. The score is evalueted from P as follow

$$Score = -10 \log_{10} P \tag{2.2}$$

Using metagenmoic libraries requires a step of quality assessment. This is fundamental in order to ensure that the data we are using are not somehow corrupted. Many tools perform some kind of quality control before starting the analysis they are made for, but usually those controls are partial and involve only the features required in the following analysis. FastQC [16] is instead a tool created specifically for the quality control of read libraries: its analysis are implemented in order to access the reliability of the sequencing process that created the read libraries.

In this situation we know fore sure that our data are not corrupted, since we simulated them. Anyway, it can be useful to analyze the report of FastQC on the simulated read libraries: it will be a good way to evaluate the performance of Camisim, which is the real source of our reads. Here, we are going to discuss the meaning of the items that are listed in the report and for all of them we will compare the result obtained from our simulated data with those presented in the FastQC manual as examples. For each of its items in the report, FastQC signals two levels of warning: those are arbitrary settings that can be modified. While using quality control tools is fundamental to have your own expectation about the data content or a minimal requisite to satisfy, weather then base quality judgement on FastQC warnings.

- Per Base Sequence Quality

  As mentioned before, the read libraries are collections of read sequences and each base of them have a quality score. The first statistic that FastQC presents is the

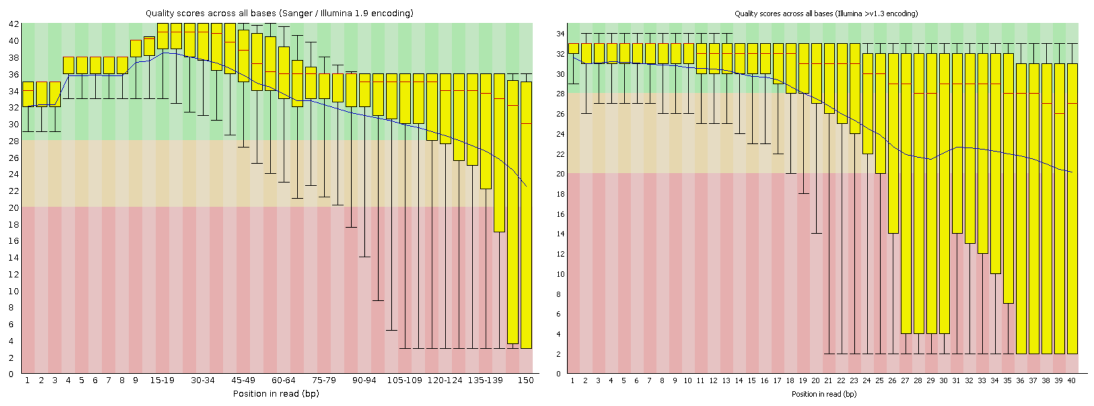


Figure 2.3: This box-plot represents the distribution of the quality scores for each base of the reads. Each column is referred to a base and contains the quantiles of the distribution of the quality scores in that base for the read library. The different colors of the background represent warnings thresholds arbitrary chosen by FastQC. On the left we have the per base quality profile obtained from our simulation, while on the left there is the one that FastQC presents as example in its manual [17].

sequence of quality distribution for the i^{th} base in the read library. Here we show an example of such information.

This plot is particularly relevant because we noticed that its values are reproduced in each of the (forward) read libraries generated by Camisim. All the backward read libraries have an other common distribution. This suggests that this distribution have been taken as model in order to generate reads simulating an illumina sampling. The profile of the distribution is coherent with what we would expect in a real case scenario: the quality is very high an the beginning of the sequencing and then falls down to critical values near the end. This behaviour is similar in all the sequencing processes and is the reason why pair end sequencing is used instead of sequencing longer reads.

- Per Sequence Quality Scores

  This statistics represents the distribution of the average quality values of the reads. In our case, the plots are very low informative, since all the reads results to have average quality values between 29 and 35, with the most abundant values being 32 for the forward libraries and 31 for the backward. Again the distributions results to be the same in all the samples with same orientation.

  In general, this is instead a very useful statistic, since highlight possible contamination of a subset of reads. Seeing a group of reads having an average quality lower then the the most abundant value might indicate that the sample have been

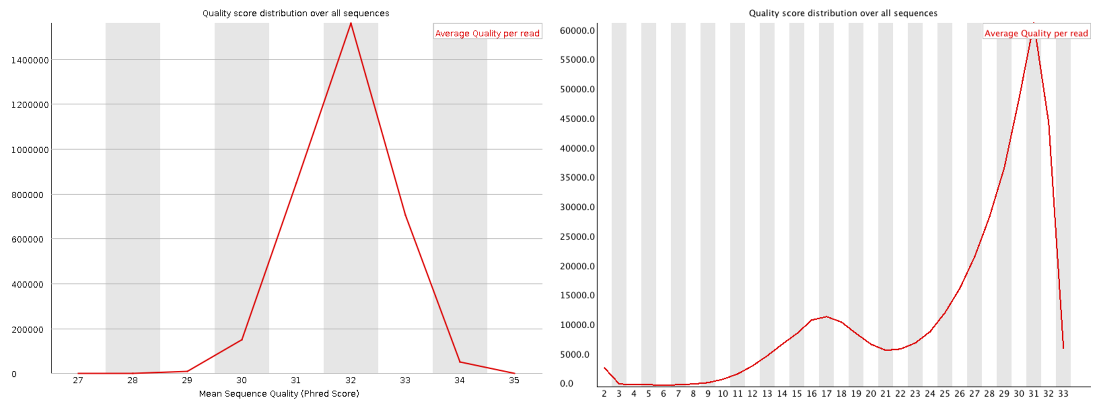


Figure 2.4: Those histograms represent the abundance of reads (Y axes) with a given average quality score (X bins). On the left, we find the histogram produced by our simulated data, where we can see that Camisim produces reads with uniform quality. On the left there is the example plot from FastQC manual; here we see a more spread distribution, with also a suspicious amount of low quality reads that might require to be filtered.

polluted or that, at a certain point in the sequencing procedure something went wrong. From this plot we can also see how abundant are the polluting reads and where eventually set a threshold in order to filter them out.

- Per Base Sequence Content

  This report takes into account the abundance of different kind of bases in each read position of the read, represented on the x axes. On the plot we have four horizontal lines that represents the different bases: "A", "T", "C" and "G"; those lines are essentially a representation of four bar plots on the same background. In the Y axes we have the percentage of that base occupying that read position in the read library.

  In our case this report is low informative, since all the bases are represented with a percentage that reflects the GC content (GC) of the original genome. The four lines are flat and for each read position can have just two values: for G and C the values is GC and for "A" and "T" in $1 - GC$.

  In a real sampling scenario, we expect to have situations in which those proportions are less uniform, mainly in the first bases of the rear. A very peaked abundance of a certain base (or sequence of bases) in a certain position might indicate that many reads have been overwritten with a certain sequence in that position. This usually happens in the first bases of the reads because of the presence of some adapters residual.

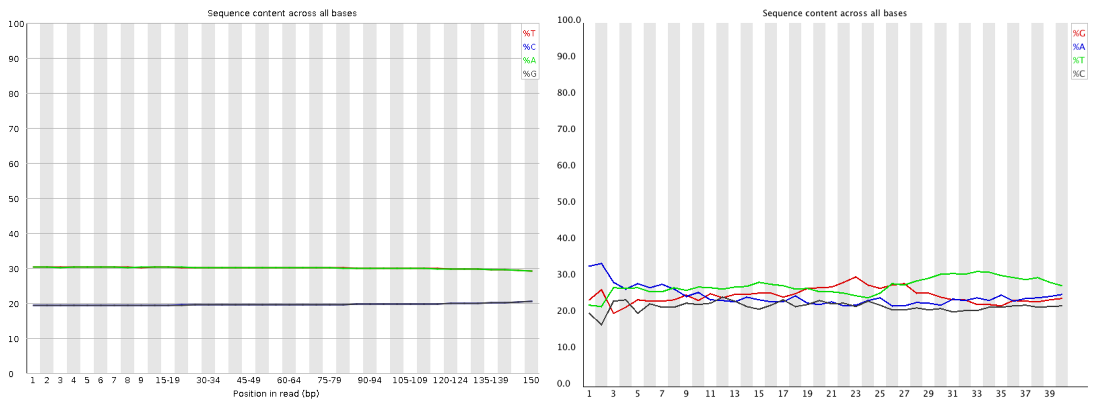


Figure 2.5: The lines in the plots are 4 histograms, representing the relative abundance of a certain symbol (Y axes) in a specific sequence position of the read (X axes). On the left we have the plot produced by the simulated data and we can see how uniform is this concentration; A and C concentrations are equal respectively to T and G, since those two are the bases that couples in the double helix structure. On the right we have a more realistic plot, found in FastQC manual: is very common to find irregular profiles of abundance, especially in the first positions of the read.

- Per Sequence GC Content

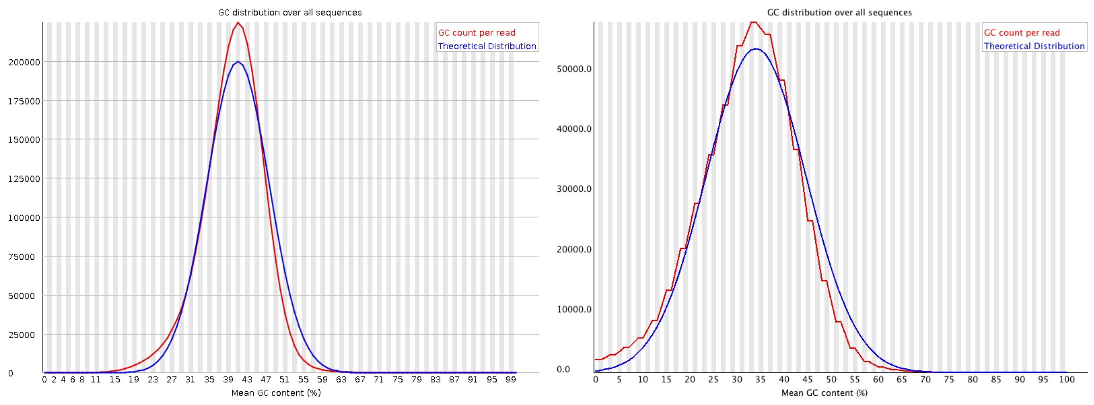


Figure 2.6: In this plots we find the real distribution of reads GC content in the sample (red) and the distribution that would be expected given the overall GC content of the sample(blue).

In this report we find an other statistics relative to the GC content. In this case, this value is referred to the reads and in the plot we find the distribution of the GC content of them (red). FastQC evaluate also the average GC content of the sample and, based on that, plots an ideal distribution of reads GC content (blue). A big divergence fot the two distributions could indicate the presence of a biased subset of reads that could be due again to some kind of corruption of the sample. The samples that we simulated resulted to have a well shaped GC content distribution.

- Per Base N Content

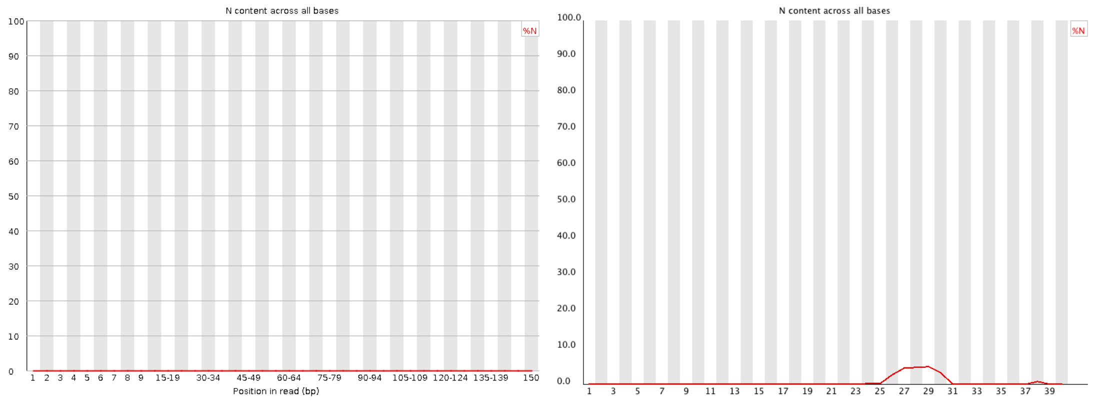


Figure 2.7: In those plots is represented the percentage of "N" symbols found in a certain read position. In the data that we simulated we do not have any "N" symbol, so the plot on the left results completely flat. In real case scenario is common to encounter some "N"s, especially in the final part of the read. This is what is shown plot on the right, taken from FastQC manual.

For reach read position, on the X axes, indicates the amount of "N" symbols in the read library. In real sampling scenarios we might expect a non 0 proportion of such symbols near the end of the read sequences. FasqQC raises warnings if there are some positions having 5% amount of "N" in the sample. In our situation we see that Camisim simulates reads with 0% "N" symbols in the sequence.

- Sequence Length Distribution

  This report plots the distribution of reads length in the sample. In a real sample, especially after trimming, we expect to have a variety of lengths and this can be used as a filtering criterion. In our study we simulated instead equal length reads, and we did not performed the trimming, since the reads did not contained adapter sequences.

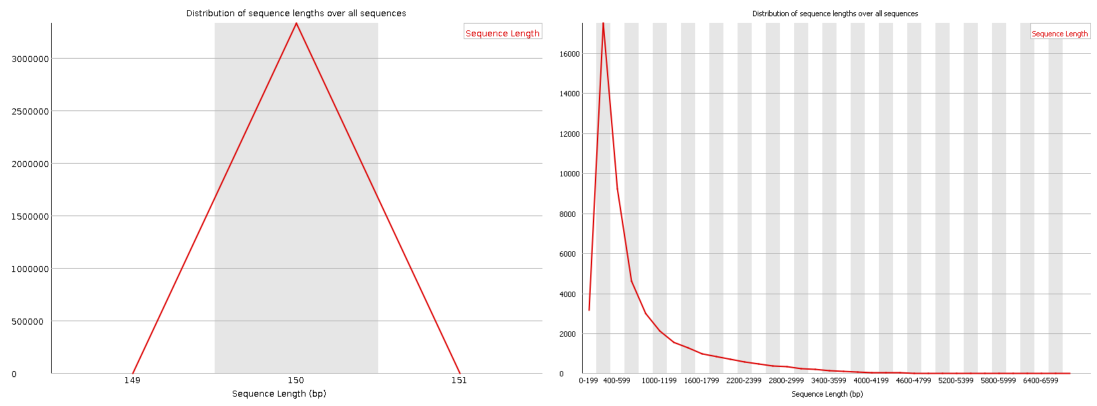


Figure 2.8: In those plots is represented the length distribution of reads in the sample. We can see how Camisim simulates reads with uniform distribution (on the left plot), while a real sample usually contains reads of various lengths (on the right, the plot taken from FastQC manual).

- Duplicate Sequences

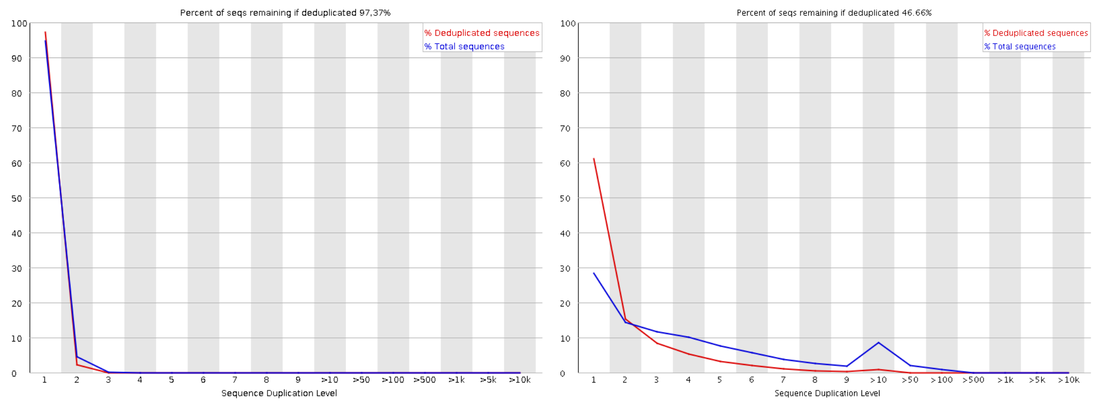


Figure 2.9: In those plots are represented the abundances of the reads with a given duplication level. This abundance is evaluated both for the original sample (blue line) and for the de-duplicated one (red). On the right we have the plot taken from FastQC manual, where we can see the presence of a sequence which is repeated a suspicious amount of times. It might indicate the presence of some kind of adapter or contaminant. On the left we have the plot produced by the data simulated by Camisim which is instead completely flat, apart for a very low amount of reads repeated 2 times.

In this report is shown the amount of repeated read sequences in the sample. In a usual sample we expect each sequence to appear only once, just in case of a very high coverage (meaning a very high amount of reads sampled from a small variety of genetic material) there might be some repeated sequences. An higher amount of repeated reads might indicate a bias in the sequencing procedure, probably in the enrichment step, or the presence of a contaminant. In the plot is shown the proportion of the library (Y axes) occupied my the reads with a certain duplication level (X axes). There are two lines on this plot: the blue one represents the proportion of reads in the original sample, while the red one represents this proportion in the de-duplicated sample. This de-duplicated sample is a sample containing all the reads in the original one with no repetitions. In the present study, the duplication levels produced by Camisim are always the same and differs just between forward and backward oriented reads. This don't seems to be dependent from the length and the sequence of the genome, although we expect to get some different repetition levels if increase the coverage (augment the read number or reduce the length of the sequenced genome).

- Over-represented Sequences and Adapter Content

  The Over-represented sequences report lists all of the sequence which occupy more than 0.1% of the total sample. Those sequences might be biologically important or might indicate some sort of contaminant or bias. FasfQC aligns those sequences to a reference database that contains a variety of possible sources for the sequence, like commonly used adapters. A match in this database might not be completely exact, but can be helpful in order to have an idea of what is going on since many adapter and contaminants are similar to each other.

  The adapter content report is similar to the previous one and have essentially the same goal. However, in this case are not the reads to be counted, but the k-mers that can be obtained from those. In the report plot we see if there is the presence of an over-represented k-mer and the percentage of it's presence at a certain position in the read (the starting position is taken into account). This plot is very useful in order to decide weather or not a library requires to be trimmed from the adapters.

  In this study, Camisim did not simulated the sequencing error called "read trough", so we did not encountered any trace of adapters. Moreover not any other over-represented sequence was encountered.

2.4 Resistome evaluation

Once we had obtained the read libraries, we can perform every kind of genomic analysis in order to access various information. In a real case scenario, the first step of the processing should always be the quality assessment of the read libraries and the trimming. The first operation provides an overview an multiple features and statistics about the read library that should highlight if the data are corrupted or anyhow damaged. The trimming is instead the research and removal of artifact sequences that could have been introduced during the sequencing by the device used. Both those operations results to be useless if applied on simulated data, so was skipped, even if those are required while dealing with real data.

Since we are interested in the AMR phenotype, the first step in the processing of the data was to perform an alignment of the reads against the metagenomic database CARD [18]. CARD is an heterogeneous database that contains data and information of various kind about antimibrobal resistance. It's elements are labelled with a controlled vocabulary, the Antibiotic Resistance Ontology (ARO), that allows a rigorous organization of the variety of elements that compose the database. In CARD can be found information about drug targets, antibiotic molecules, drug classes and the molecular mechanisms of resistance, but our main interest in this database is about the sequences of the genes that have been discovered to encode a mechanism of antibiotic resistance. Those are called *resistomes*. CARD contains hundreds of resistance genes and, in order to put together such an amount of information, was required a massive work by expert physicians that could not be automated. The curation of this kind of databases is a perpetual process and implies the review of all the scientific literature on the topic. Each article can determine an addition to the database or a modification of an existing resistance mechanism, if the publication satisfies the CARD curation paradigm. The resistance mechanism must be described in a peer-reviewed scientific publication, the DNA sequence associated to it must be available in the GenBank, including clear experimental evidence of elevated minimum inhibitory concentration (MIC). Moreover, CARD provides an other set of resistomes in CARD's Resistomes and Variants data-set (nicknamed wildCARD); those differs by the resistomes in CARD because are silico predicted allelic variants, determined to be reistomes by computational methods. It is highly recommended to use also this extension of the database as the allelic diversity for AMR genes is greatly unrepresented in the published literature.

CARD provides also RGI (Resistence Gene Identifier), a command line tool that can be used to analyze genomic data of various kind and aligning them to the genes present in the database. RGI can deal with assembled genomes or contigs, but also with read libraries and in this study was exploited this last modality. This procedure relies on aligning the reads to the reference genomes contained in CARD and wildCARD, but in order to perform the alignment in reasonable times the references are transformed using the Burrows–Wheeler transform. This is a reversible transformation that is widely used in data compression but also allows to query the data more efficiently. The construction of the Burrows–Wheeler transform is performed separately and one times for all the future uses, so it do not slow down the analysis. Both this operation and the alignment are implemented in Bowtie2 [19], a widely used software in the field of aligners. RGI provides a wrapper around it, specific for the CARD database and creates an output that contains all CARD information about the references that results at least in a match.

For reach reference of CARD/wildCARD that appears in the output, are provided a lot of information and also the number of reads matching the reference. From this data we can evaluate an estimate for the abundance of the genome material associated to the reference that is in the population. The unit of measure of such abundance is FPKB (fragments per kilo-bases per million reads). This measure is evaluated for each reference gene the reads that are aligned to: the count of aligned reads is scaled for both the length of the sequence (we expect a linear dependence of the number of matches and the number reference length) and the number of reads in the sample, in order to make the abundance value comparable with the results from samples with different read counts.

$$FPKB = \frac{Aligned\_Reads}{Reference\_Length * Reads\_in\_the\_Sample} \tag{2.3}$$

In order to get the usual scale of FPKB, the result should be multiplied by $10^3 * 10^6$. In this way, we obtain what we can call a *resistance spectrum*: a multidimensional quantity that represent the abundance of different resistance sequences in the genetic material that was sequenced. It can be visualized as a bar plot and it contains an entry for each different ARO identity identified in the sample. We can call *resistance spectrum* the full set of resistomes taken into account. Since we needed to compare the spectrum obtained by various samples, the actual ARO domain that was used was the union of all the domains of the samples. The collection of those spectra usually results in a very sparese data-frame that will be the starting point for our machine learning procedures. In the following image 2.10 we present a bar plot representation of the resistance spectrum that can be obtained with this procedure. For sake of visualization, the image will represent a version of it which have already been filtered, leaving out the majority of columns. Those are usually are in the order of 1000 ARO identities, but most of them have very low abundance can't be exploited in the following analysis. A useful statistic to understand the loss of information in the filtering process is what we can call the *resistome integral.* It can be defines as the sum of all the abundances of a spectrum and its measure is still FPKB. During the filtering operations that we perform, the integral is usually reduced of its 10% loosing the 90% of the ARO identities. Reducing of such amount the dimension of our spectrum is important in order to speed up the computation and but we tried not to decrease the *resistome integral*: a reasonable trade off have been chosen.

The distribution of the *resistance spectrum*, the *resistome domain* and the *resistome integral* strongly depends on the content of the metagenomic sample, but also on the database used as reference for the alignment. Moreover, the *resistome domain* depends also on the other samples and the other spectra we are considering, since it have to describe the full set of samples we are going to analyze.

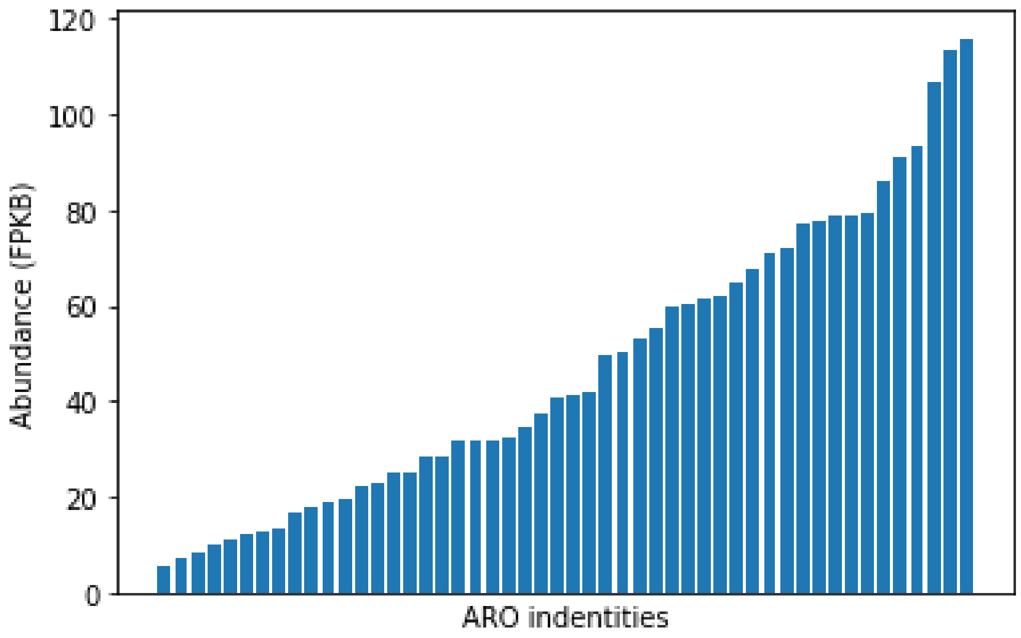


Figure 2.10: Each column represents a different ARO identity found during the alignment process in one of the populations. For sake of visualization, we plotted the resistance spectrum already filtered (see chapter 2.4 for the filter procedure); in this case we are left just with 51 columns over the original 1335, while decreasing the spectrum integral of its 11%

2.5 Preprocessing and PCA

As mentioned in the previous section, the obtained data are very high dimensional. In a simulated sample we found tracks of hundreds of different *resistomes* and we also need to add the other *resistomes* that are found in other samples; in the one we are considering will be counted 0. In order to train a machine algorithm with such complex data would be required an absurd amount of information and computational time. We thus needed to simplify those data and here is the list of operations that were performed on them. The starting point was, as mentioned, a 2D data-frame which has a row for each sample and a column for each ARO identity in the ARO domain. In each cell is reported the abundance of the *resistome* in the sample (FPKB). The dataframe was implemented with the python library Pandas (version 1.3.4)[20] and also all the filtering operation were performed tanks to the functions of this library. In the following list we will not specify any parameter for the filtering procedures, since those were adapted to the various samples simulated to obtain the best trade off in terms of computational times and information loss. Notice that the action of sparsity filter and of the median filter (that we are going to discuss) reduces the dimension of the *resistance domain.* This operation have to be performed being careful of not unbalance the information content of the different samples.

- SPARSITY: A *resistome* that appears in in just few samples increases the dimension of our database without bringing much information. We chose to eliminate all the columns that was showing a 0 abundance in more than the $X\%$ of the samples (sparsity $> X\%$). Notice that in each set of sample simulation we simulated 50% of an AMR phenotype and 50% of the other. We will get in deeper detail about what we mean here by phenotype, but, given this proportion, we had to keep all the columns with sparsity $< 50\%$ and also keep a wide margin over 50%. Features with sparsity $> X - 50\%$ and $< X\%$ could be indeed very good indicator of our phenotype, as they could be present in one of the groups and absent in the other. During the trials of this study this filter caused some problems in the processing of a particular design. We considered different samples, some of them (say $Y\%, Y > X$) with an high percentage of individuals similar between the samples and some other samples with different "outsider" species. In this case the sparsity filter action eliminated almost completely the resistance spectrum of the outsider samples, since many of its entry where sparse in our data although being truly informative. In order to avoid this loss of information is useful to keep in check the *resistome integral* of the samples before and after the filtering.
- MEDIAN: There are a lot of *resistomes* that have very low abundance in the samples, but still are present in many of them. Those won't be affected by the sparsity filter, but still are of very little use in our classification process. A *resistome* like this could be some sort of background noise that could be produced by a bias in the simulation or in the alignment procedure. In order to get read of such *resistomes*, was calculated the median of their abundances in the samples and those that resulted under a certain soil was eliminated.
- STANDARDIZATION: Standardization is not a filtering process, but still is a necessary step in order to make the data available for PCA. This operation is quite common and in this context we relied on a pre-implemented function of SKLearn library. The abundances of each resitome are scaled and shifted so that the average of their distribution becomes 0 and the standard deviation becomes 1. In this way all the *resistomes* that was not filtered out comes to have the same "importnace", even if those presented different abundances and different variances.

Up to now we have transformed the database just eliminating some of the columns and keeping the others as they are, a part for standardization; at the end of the filtering we expect to have a database with less then 200 columns. Since in all the simulation set up we produced far less samples, those dimensions results to be still too many for our purpose. Having data for each sample of higher dimension of the number of samples itself results in the under-determination of our problem. If we train a machine learning algorithm with data like this, we will obtain an operative classifier, but ti will just work exploiting few of the dimensions of the data. The number of them will be, at most, equal to the number of samples. In this situation, the best we can do is to identify the dimensions of our data that are more relevant if we intend to perform classification. Those dimensions won't be the original ARO identities of our starting database, but will be a linear combination of them. At this point we are allowed to cancel all the other non informative components and train our machine learning algorithms just on the N-1 most important, where N is the number of the samples we have. The procedure that we are talking about is called Principal Components Analysis (PCA) and is implemented in a function of the Scikit learn python library. To perform this procedure, the data needs to be centered, so that the average of all their components (column values) have to be 0. Their variance have to be the same, otherwise the PCA will privilege the components with bigger variance, even if it is due to a bigger abundance of the component. If the data are in this condition, PCA can be correctly applied and its results is a new database with lower dimension of the original one: just N-1 columns remains. The data are represented in a different base which is a linear combination of the original columns. The new base has its columns sorted by the PCA in order of "variance explained". This expression refers to a parameter that is evaluated by PCA and is relative to the amount of variation that the data present if projected on that component. For a better understanding of the meaning of the explained variance and PCA, see section 14.5 of [21]. Based on this parameter, the algorithm sorts the new columns of the database and then we can get read of the less variance explaining components an keep just the N-1 most relevant.

2.6 Machine learning

The complexity of our data has been reduced a lot. The filtering operation we performed resulted in a considerable loss of information, but due to the limited number of samples, we could not really take advantage of it in the classification process. With PCA we managed to identify the most relevant combinations of components and we kept just them. We find ourselves with a quite smaller 2D database, composed by N rows and N columns. It is possible now to use the machine learning algorithms that we considered in this study. Although being different in their implementation, the ML algorithms can be treated quite in a similar way and we will start this section with some general considerations about them (subsections 2.5.1, 2.5.2, 2.5.3).

After this, we will give a brief explanation of how the ML algorithms used in this study work. All of them are used for classifying multidimensional data that can be described in a multidimensional feature space. Some of the components of such space might be continuum variables, other might be categorical. In the case of a ML algorithm used for classification, we have one additional categorical variable which will be the object of the guesses of the algorithm. In this study, the categorical variable will have just two possible values: "resistant" or "susceptible". Like in all the ML process, the classifiers have to be trained using a set of data in which we know all the features and ad also their classification. Understanding how the training process tune the parameters of the ML algorithm is the base for understanding their functioning.

The classifier that was exploited for this analysis was:

- Elastic net logistic regression (subsection 2.5.4)
- Random forest (subsection 2.5.5)
- AdaBoost classifier (subsection 2.5.6)

The implementation of them an them can be found in the following python libraries:

- sklearn.linear_model.LogisticRegression for the elastic net logistic regression
- sklearn.ensemble.RandomForestClassifier for random forest
- sklearn.ensemble.AdaBoostClassifier for AdaBoost classifier

All the parameters have been left with default values, apart from those that we will mention in the following sections.

2.6.1 Phenotype definition

In first place, we have to provide a suitable definition of phenotype. Up to now we have mentioned this term, but we did not explain what is its exact meaning in this study. The National Human Genome Research Genome Institute defines it as "the individual's observable traits, such as height, eye color and blood type". It is reasonable to include in the list of observable traits also the resistance or susceptibility to a particular antimicrobial substance. An aspect of this definition that instead is not compatible with this study is the fact that the phenotype is referred to an individual. Referring the phenotype just to an individual can be indeed problematic when we are dealing with metagenomic data obtained by wild bacteria populations. In this case, the samples contains information about a huge variety of individuals, some of them may be resistant and some of them may be not.

One possible solution to this problem could be to perform a taxonomic classification of the genomic material of our samples. This is a well established routine and it results in the classification of the reads based on the specie of their source. In this way we obtain a set of sub-samples, each of them referred to a homogeneous population (more or less, depending on the taxonomic classification parameters). We could now assume those populations to be made of exactly equal individuals; in this case the resistance spectrum that we obtain from the taxonomic sub-set of reads will be informative on each single individual of the taxa. From those spectrum, which are now referred to singular individuals, we could evaluate the phenotype. In this way we have recovered the singularity of the phenotype definition, but the assumption that we made was very strong. Whatever the granularity of the taxonomic classification is, we end up considering the bacteria population as made just of few groups of bacteria, each group made of cloned individuals. This simplification becomes less problematic as we increase the resolution of our taxonomic classification, but we can't increase it too much. Indeed, given the finite sample size, if we divide the same number of reads in a bigger number of species, we end up with a smaller number of reads per specie and this is problematic if we intend to evaluate the resistance spectrum with an appropriate depth.

The solution we adopted was another one: we provided a more flexible definition of phenotype that fits the purpose of our study. The phenotype will be defined as a metadata associated to a bacteria population. In this study, the phenotype will be a binary data that can have the values of ”resistant” and ”susceptible”. Depending on the population design, the meaning of those two values will be defined more clearly, but we can anticipate some examples:

- In populations produced by a single specie, so produced by a Camisim run that had just a single genome as input (it will contain just the strains simulated from it), the phenotype of the population is considered to be the phenotype of the genome. The genome phenotype is the metadata provided by PATRIC and it is the result of a laboratory experiment on AMR phenotype made on populations of ”cloned” individual, all having that genome. The experiment can be disk diffusion or MIC evaluation, but we will refer just to the ”resistant”/”susceptible” data.

- We also simulated populations of 4 species. In all the simulations 3 of the 4 species was kept the same, and the phenotype of their genome was ”susceptible”. The fourth specie was changed each simulation and half of the times its phenotype was ”susceptible” and half of the times its phenotype was ”resistant”. In this design, the population phenotype is ”resistant” if there is at least a specie simulated by a ”resistant” genome, otherwise the population is ”susceptible”.

Those definitions might appear not intuitive and the fact that the phenotype have to be defined again for every kind of population can be problematic. Never the less, those definitions are useful if we use them for machine learning; those indeed provides a clear way to classify the populations that was simulated. If we train a ML algorithm with the resistance spectrum of various populations of 4 species and their phenotype as defined it in the second example, the algorithm will be able to make predictions on other samples. Those predictions will be in therms of ”resistant” or ”susceptible” as we defined them in the training set. This means it will be able to distinguish populations of all "susceptible" species to populations with at least one "resistant" specie on a constant background. Notice that a ML algorithm trained as described will produce nonsense prediction on a population with more then one "resistant" specie, since it is outside the domain of the phenotype definition it was trained for.

In the Results section, will be specified the definitions used for each population design that was simulated. Anyway, we can anticipate that all of them are boolean metadata. This is due to the data availability on PATRIC, which provides boolean information about the AMR phenotype in the majority of cases. Never the less could be considered the possibility of repeating this kind of study taking into account the MIC, weather then the categorical definition of phenotype. MIC, being a positive real number, is clearly more informative then a boolean variable.

2.6.2 Training, validating and testing

Before getting in the detail of how the machine learning algorithms are implemented, we should consider the structure of a ML study, which is independent to the classifier used. ML classifiers are procedures that are capable of assigning a labels (in our case, the AMR phenotype) to multidimensional data (in our case, the resistance spectrum). In order to do this, those algorithms are trained on a set of data which have already their labels assigned. The training procedure is specific for each kind of classifier, but in general we can say that the algorithm finds the best set of internal parameters in order to succeed in the classification of known label data. This is usually implemented as the minimization of an error function that is achieved with an iterative process. The more we train the ML algorithm, the more it will be efficient in assigning the labels to the training data-set. In this process we have to keep in mind that the training will improve the model performance just on the training data, while instead we usually want to have good performance on all data of that kind. An excessive amount of training (amount of iterations) will end up in the so called over-fitting the training data. In this condition, the classifier have very good performance on the training data-set, but scars performance on other unknown data.

In order to avoid over-fitting and test the result of the training procedure, the performance of a ML classifier are evaluated on a set of data which was not used in the training process; this practice is called validation. An usual measure of those performance are the ratio of correctly classified data over the full validation data-set: we will call it *score*. Also counting the type of errors made is a useful statistic (classified "susceptible" while being "resistant" or the opposite). Notice that the validation data-set have to be structurally equal to the train one: even if the labels of the first are non provided to the classifier, we need to know them, in order to understand the validation results. In practice, this means that, if we intend to perform a ML procedure on a data-set, we need at least to divide it in a train and a validation sub-set.

There are a variety of ways to do so, but the more basic one is simply to chose a proportion $0 < P < 1$ and randomly extract $P * data - set.size$ elements form the data-set and use them as validation set. Notice that, if we change the entries that we move to the validation set, the training of the classifier will end up in a different way. This because the values of the internal parameters of the classifier are determined by the training set and different training sets produce different classifiers. This means that, even if we establish a proportion P for a split of a given data-set, we have a variety of possibility to chose the $P * data - set.size$ validation entries and for each of them we will have a different classifier at the end.

This under-determination is clearly problematic, but provides also a more detailed picture of the performances of the ML procedure we are using. In order to capture the results from a variety of possible train-validation split, we exploited various cross validation techniques. A k-cross validation procedure consist in dividing the data-set in k equal size folds, rather then in just 2. A classifier is trained on k - 1 folds and its performances are validated with the remaining fold. This procedure is repeated k times, using a different validation fold each time. The collection of scores (and the errors type) for all the validations performed is a good indicator of the performance of our ML procedure. The data we will report in chapter 3 will be the average and the variance of the scores distribution. In this study, we used a five fold cross validation ($k = 5$) and a live one out cross validation($k = data - set.size$). This kind of cross validation techniques have the peculiarity of including each sample in the validation set exactly one time. This implies that, for each sample, we have exactly one prediction at the end of the validation process. Since we intend also to report which kind of error committed the ML algorithm (false "susceptible" or false "resistant"), we will refer to this prediction.

In some studies is required to chose, between all the k classifiers trained in the cross validation, the one that best performs. In this case is required a supplemental set of data in order to perform a testing of it independent from the data used to create it. We can't indeed relay on the score that the classifier performed on the cross validation set, because the choice of the classifier is determined from the validation set itself. In the present study we did not performed any kind of test, since our purpose is not to obtain the best possible classifier, but just to have a picture of how ML classification works on the data we simulated.

2.6.3 Feature importance

In many situations, referring to machine learning algorithms, those are said to be similar to "black boxes" that are able to perform classification from the features of the input data. This wording refers to the fact that the way in which the features are processed is quite complex but don't have to be known in order to exploit the classification obtained. This kind of approach, although being useful in naive experiments, is quite poor and keeps the user away from a better understanding of the content of its data.

A first step into those "black boxes" understanding is to evaluate the feature importance that results from the machine learning training. There are various measure of feature importance, but all of them intend to quantify how much a feature results to be relevant in the classification process. In this context, the features that are provided to the classifiers are the components evaluated from the PCA: linear combinations of the resistance gene abundances that are found in the metagenomic samples during the alignment. We adopted this strategy for computational reasons, in order to reduce the dimension of each sample and the computational times, but in principle those features could be the *resistome* abundances themselves. Evaluating the feature importance for the *resistomes* means to access how those are relevant in the determination of the population phenotype. This is quite an important aspect of the analysis that might enable to link the result of the machine learning classifiers to biomedical knowledge about the properties of the *resistomes*. Moreover, comparing the various importance values obtained from different classifiers will be informative about the differences in the resulting phenotype.

In the following subsections, we will explain how work the classifiers that we used and we will also clarify what is the importance measure that we took into account. Notice that feature importance values are not supposed to be comparable, since are defined differently. Anyway, the order of the features sorted by those importance values can be instead object of comparisons.

We recall that in this study the classification of the samples is achieved with cross validation strategies. This means that, for each classifiers used, we trained various copies of it on different training sets. As a result, those classifiers will assign different importance values to the features of the data. In order to provide a detailed picture of the actual importance of a feature, we will present the average values and the standard deviation of the importance values of each classifier trained during the cross validation.

2.6.4 Elastic net logistic regression

Logistic regression is the first classifier we have taken into account. This model belong to the category of generalized linear model and indeed the equation that we are fitting to the data can be obtained from a linear fit. Generalized linear models are fitted to the data maximizing the likelihood (likelihood of those data being sampled from the candidate distribution). Before getting in the detail of the fitting procedure, let us present the visualization 2.11 of the procedure result of it for the logistic function: we will start with an heuristics analysis of it.

The equation 2.4 that we are fitting is the following, where the parameters will be called slope (k) and intercept (x_0):

$$Y = \frac{1}{1 + e^{\mathbf{k}(\mathbf{x_0} - \mathbf{X})}} \tag{2.4}$$

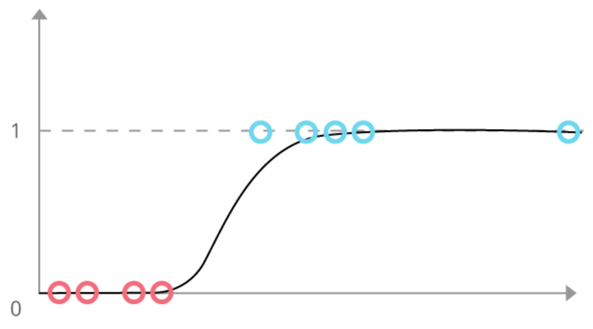


Figure 2.11: In this picture we see a plot of a logistic function fitted to the data, which are represented by the circles. The different colors of the circle represent different classification of the represented. The horizontal axe represent the domain of the function and in which are described the features of the data. The co-domain of the function goes from 0 to 1 and represents the probability of a data to belong to one of the two category. The training data are placed at the boundary of this domain, since their classification is secure

While the intercept have to be bigger then 0, there is no limitations for the slope values: the absolute value of this second parameter is proportional to the slope of the curve and if it is negative, the curve results up side down. The domain of this equation is the all real number set, while the co-domain is limited to the interval $[0-1]$. In our case, the dimension of our data is bigger then one, so this picture results to be actually a projection of a multidimensional curve which associates values of a N dimensional domain (R^N) to the co-domain interval. In various context a probabilistic interpretation is associated to the co-domain of the logistic function, but, in our case, we will use it for classification porous. This means that the co-domain is split in two sub intervals which will be associated to the catheterise we are dealing with: $[0-0.5]$ and $[0.5-1]$.

The data which are already classified are represented in the plot by the dots of different colors. Referring to the probabilistic interpretation of the logistic function, those are represented at the boundaries of the co-domain, since we are sure about their classification. Fitting the values to of slope and intercept to the known data allows us to exploit their information in order to have a guess about new data classification. This is actually an elemental form of machine learning, in the case the calculation is performed by a computer. When the best fitting curve is found, new data with unknown classification can be taken in to account and the image of them will be informative about the probability of belonging to a category or an other. We assign to the data the more probable label. Notice that, because of the shape of the logistic function, the domain which is associated to an intermediate probability (say [20% to 80%]) is actually quite small respect to the rest of the domain, where we have an high probability for the element to belong to one of the categories. This is why logistic regression is used also as a classification method, assigning the elements of its domain to the most probable category.

In this context, the slope of the curve is very relevant, since the steeper it will be, the more sharp will be the classification. In the context of multiple dimension domain, we have a slope coefficient associated to each of the variables in the scalar product that we have at the exponential. The k components (it's absolute value) will then be used as a measure of the feature importance for the features of the domain they are related to.

In this context, we did not used this exact equation, but a regularized version of it. Regularization is an usual procedure in statistics and is used in order to reduce the sensibility of the fitting algorithms to the training data. This is useful in order to avoid over-fitting the data and can also be helpful in reducing the dimension of our problem. In a moment we will explain the fitting procedure and the regularization, but we can anticipate that the regularization affect the value of the slope, reducing the importance of the features. One of the regularization techniques that we use, the Lasso regularization, is capable of putting some components of the slope to 0, making the variable useless in the classification and reducing the dimension of the problem.

Let us now consider the negative log-likelihood of a logistic distribution P in the light of a set of known data $\{x_i\}$. For each x_i, y_i will have a value between $\{0, 1\}$ depending on the category they belong to 3.3.

$$P_i = P(\mathbf{x_i}) = \frac{1}{1 - e^{-\mathbf{k}(\mathbf{x_i} - \mathbf{x_0})}} \tag{2.5}$$

$$y_i \in \{0, 1\} \tag{2.6}$$

$$L_{nlog} = -\log(L) = -\sum_{i=1}^{N}[\log(1 - P_i) + y_i \log(\frac{P_i}{1 - P_i})] \tag{2.7}$$

If the we explicit the logistic distribution as a function of x_i and the other parameters, we get 2.8:

$$L_{nlog} = -\sum_{i=1}^{N}[-\log(1 + e^{\mathbf{k}(\mathbf{x_0} - \mathbf{x})}) + y_i * \mathbf{k} * (\mathbf{x_0} + \mathbf{x})] \tag{2.8}$$

Minimizing this equation results in finding the best values for the parameters slope and intercept. From this formulation, it is easy to introduce the regularization terms we mentioned before. Since we aim to reduce the sensibility of our fit by reducing the values of k, it will be enough to penalize bigger (absolute) values of it in the L_{nlog} function. There are two ways of doing so: Ridge and Lasso penalization. In this study we combined them with equal proportion in order to have both the advantages of the two techniques: this hybrid regularization is called elastic net. The regularized negative log likelihood takes the following form 2.11:

$$R_{Ridge} = \sum_{j=1}^{N} (\mathbf{k} * \mathbf{x_j})^2 \tag{2.9}$$

$$R_{Lasso} = \sum_{j=1}^{N} |\mathbf{k} * \mathbf{x_j}| \tag{2.10}$$

$$Reg\_L_{nlog} = L_{nlog} + \lambda[\gamma * R_{Ridge} + (1 - \gamma) * R_{Lasso}] \tag{2.11}$$

Λ is a parameter that regulates the intensity of the penalization: it can go from 0 (normal logistic regression) to potentially ∞ for stronger ad stronger penalization. Γ's domain is instead limited between 0 and 1 and it determines the proportion between the two types of penalization.

The two penalization terms have similar structure: both aims to reduce the norm of the k vector. Ridge penalization is implemented referring to the L2 norm of the vector and results in a very strong penalization for high values of k. Lasso penalization is instead referred to the L1 norm of the vector: this means that the penalization is linear (and not quadratic) respect to the vector length. This results in the fact that, although Lasso regression will penalize in equal proportion big and small vectors, it is capable of penalizing them to zero if the combination $\lambda * (1 - \gamma)$ is sufficiently big. If a component of the slope vector results to be 0, we end up neglecting the relative component of our domain and we achieve a dimension reduction.

2.6.5 Random forest

A random forest is a machine learning method which is used for classification, but also for regression and other tasks. In this subsection we will just consider the first use, which is the one which is exploited in this study. Random forests are called ensemble machine learning methods, since the classification they perform is the result of the average of the result of multiple classifiers. Those are trained with data-sets obtained by the original training data-set with various techniques (boostrap). In the case of a random forest, the classifiers that compose the ensemble are classification trees and the final classification of the input is decided by the majority of them. In order to understand random forests is then important to consider how classification trees work.

A decision tree can be represented as a set of splits in the feature-space in which the data are describes. Those splits are parallel to the axes and the regions that they define are hyper-rectangles. Each rectangle is assigned to a category, and the data that falls in that rectangle are categorized in that way. In the training process, we define the position and the shape of the splits that we perform on the feature-space and the category assigned to the hyper-rectangles so that those are able to correctly classify the training data.

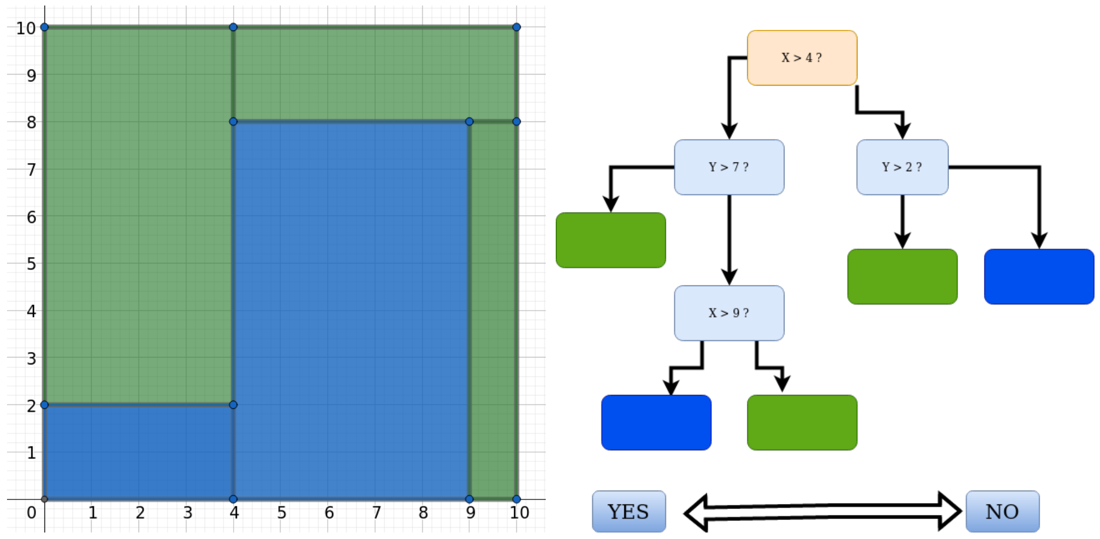


Figure 2.12: In the images we have the two representation of a trained decision tree that acts on a two dimensional domain $[0-10]*[0-10]$. At the left we can see how the domain results divided from the splits on the two variables. The colors corresponds to the two categories that the decision tree is working on. The same coloration is reported in the leaves of the network that can be found in the image at the right. In this tree network we have a root node, in light orange on the top, and three branches, in light blue. Each of them corresponds to a split in the domain based on the condition written in the node. The bottom arrow indicates a formalism which is often used in the decision tree representation: between the two outgoing arrows, the one pointing to the left link node that respect the condition in the starting node, while the node pointed by the left-going arrow does not.

The algorithm that is used to do so can be well explained if we represent the splits as the nodes of a network. This network have specific properties and nomenclature:

- The network is a directed tree: each node of the tree might have many outgoing edges, but must have at most one ingoing edge.
- In a classification tree, each node have at most two outgoing edges.
- The network is connected and the only node with no ingoing edges is called root node.
- The nodes which have no outgoing nodes are called leaves.
- The nodes which don't belong to the previous categories are. called branches

In the training process we start considering just one of the multiple features of our data and a split value for that. This feature might be categorical, in this case we will be splitting a discrete domain in two sets, or continuum and ordered, in this case the split domain using a threshold. To decide how to perform this split, we will consider the purity of the two resulting sets of training data. There is a variety of purity measures, but in our study we used the weighted sum of the Gini Impurity 2.12 of the resulting set: we will tray to minimize this value in order to obtain the best possible purity from our split. Let us consider a tree that needs to classify the data between resistant and susceptible category. Being R_i the number of resistant type element in a set and S_i the number of susceptible type element in the i^{th} resulting leaves:

$$G_i = 1 - (\frac{r_i}{r_i + s_i})^2 - (\frac{s_i}{r_i + s_i})^2 \tag{2.12}$$

$$Purity = -\sum_{n=1}^{2}(G_i * (s_i + r_i)) \tag{2.13}$$

The highest purity split is chosen for that component of the feature space. For each component is repeated this process until we find all the possible highest purity split. At this point is chosen the component which have the split with the maximum purity and the training data-set is divided based on the split on that component. With this procedure, we evaluated the direction and the value of the first split of our feature space; it is represented by the root node.

Whatever have been the chose split, is unlikely the Gini impurity to be 0 in both the resulting subsets (the minimum value for the Gini impurity is 0). In order to increase the precision of our classification, we will perform other splits on the two subsets of data that was obtained from the first split. Those new splits will probably be on other variables and with other values: those will be determined with the exact same procedure described before. Those splits are represented as the branch node of the tree. If we find ourselves with a subset of data composed just by resistant or susceptible elements, we will stop splitting it, since it's already pure. Those data are located in a hyper-rectangle that will not be partitioned any more and that will be represented as a leave. We need to assign to all the leaves a type between resistant and susceptible in order to provide a meaning for our classification. In case of a pure leave this is straightforward: the category of a pure leave is the category of the training data that falls inside it.

If we let this iterative process go without any limit, we will end up with just pure leaves. The classification tree will have achieved the best possible precision in classifying the training data. Anyway, as many machine learning procedures, if we train too much a classifier on a data-set, the classifier will end up over-fitting it. This means that it will have very good performance in classifying the training data, but will not be able to classify new data of the same kind. For this reason, usually we set a limit for the leaves Gini impurity which is bigger then 0 for stopping the splitting. In this case we end up with impure leaves, which represent subsets of feature-space containing both resistant and susceptible training data. In order to chose a category for those leaves we can consider various strategies, but usually we simply assign to the the category of the most abundant training data that falls into them.

For a decision three classifier, the feature importance is determined by how much the best split on that feature reduces the impurity measure of the resulting leaves. This value is then normalized in order its sum over all the features equals 1.

Now that we have trained the classification tree, we can use it to classify new data. Those will be represented in the same features space that we have partitioned and will be located inside one of the leaves (hyper-rectangles) that we defined. The classification performed by the classification tree results in assigning to the new data the category of the leave it falls in.

Random forests are a collection of multiple decision trees (100 in the present case): each of them classify the same input data and the results are averaged over all the trees. This kind of machine learning algorithms is called ensemble, since is based on the average of the results of multiple classifiers. Considering many classifiers, rather then a single one, makes sense only if those classifiers are different to each other. The threes are created always with the same criterion, so in order to differentiate them we must have different training data. In order to train a random forest we indeed obtain various training data-set from the original one using a bootstrap procedure. In this way we get many training data-sets of equal size, containing some repeated data and without some other; different trees will be trained on those data and those will compose the random forest.

In the training of those trees, at each node, the choice of the feature on which perform the split will be limited to random subset of the total features: just N of the total will be considered. The performance of those trees is evaluated on the data left out from the training set during the bootstrap. This allows to chose the best value for N.

The procedure described aims to diversify the tree created while taking information from the same training set, which is the one provided to train the random forest.

The feature importance that is evaluated from a random forest results to be the average of all the importance that the decision threes assign to the features. A decision three is taken into account in this average only if it actually contained this feature in its training data-set. The resulting values ate then normalized again.

For a better understanding of this subject, we suggest to read chapter 9.2 of [21].

2.6.6 AdaBoost Classifier

The AdaBoost classifier [22] is an ensemble machine learning technique as well as the random forest. Also in this case a set of simple classifiers is used to evaluate different guesses and then the final result of the classification is obtained taking into account all of them. In this case the classifiers shape have relatively low importance and usually the simplest possible structure is chosen. The structure used in our study is called stomps and can be described as a three node tree: one root and two leaves. Respect to the random forest, the AdaBoost classifier simplify the structure of the classifiers, making them all equal to each other. On the other end, AdaBoost construct those classifiers in a more refined way, taking into account the results of the classifiers already constructed. Moreover it assign difference importance to the results of those classifiers when is evaluating the final result.

A stomp can take into account only one component of the multidimensional feature space that is used to describe the data. AdaBoost procedure uses the same strategy explained for the random forest in order to identify which is the component where the split should be made and at what value, in order to get the maximum purity in the resulting leaves. With this very simple split of our feature space, we can perform a classification of the training data. The purity of this result will determine how much the stomp will be taken into account when a prediction on new data will be performed. This feature of the sub-classifier is called "say" and can be evaluated like this 2.14:

$$Say = 0.5 * \log \frac{1 - Error}{Error} \tag{2.14}$$

For the first stomp, the *Error* therm in that expression can be considered just the number of miss-classified over the total number of them. Few things should be said about the shape of this function:

- The say can be negative. This happens when we have a sub-classifier that makes more mistakes then correct predictions. In this case, multiplying it's guess for a negative value will reverse the prediction when we are evaluate the final average guess.

- If a stomp perfectly classifies the samples ($Error = 0$ or $Error = 1$) the say diverges, so in practice a small amount of Error is always added or neglected.
- A stomp gets the lowest possible consideration in the final evaluation if can't be found any correlation between it's guess and the real classification of the samples. More specifically, when half of samples are correctly classified and the other half are not.

With this function we can determine the importance of a stomp based in its performance on the training data-set. AdaBoost classifiers exploit the result of the first stomp to update the training data-set, in order to give more importance to the miss-classified samples. To each of the sample is indeed assigned a weight such that the sum of all the weights equals 1. In the case of the first stomp the weight are uniformly assigned, but after the first prediction is evaluated, the weights are updated in the following way [2.15 2.16]:

- For miss-classified samples we have

$$New\_weight = weight * e^{classifier\_say} \tag{2.15}$$

- For well-classified samples we have

$$New\_weight = weight * e^{-classifier\_say} \tag{2.16}$$

The new weights are then normalized in order to have sum equal 1. In this way we have essentially obtained a different training data-set and so we can construct a new classifier following the exact same procedure. The new classifier (a new stomp) will differ from the first since also the training data-set is different. More precisely, when we are looking for the best dimension on which perform the split and the best value to do it, we should refer to the weighted definition of the Gini impurity index, that takes into account the updated weights.

Alternatively, we can sample a new training data-set from the original one. The sampling will allows repetition of sample choice and will end up in when the new data-set have the same size of the original one. The sample should be a stochastic process that follows the probability distribution that is specified by the samples weights. The two strategies are not exactly equivalent, but usually, and also in our study, is used the second, since is computationally more efficient.

When a desired number of stomps (50 in our case) have been trained and their say have been evaluated, the AdaBoost classifier is considered to be trained. As anticipated, the classification performed by this tool will be the result of a weighted average of each stomp result, where the weights of the classifiers are the normalized says.

Also the features importance will be the weighted average of all the importance values obtained from the stomps that actually take that feature into account. Notice that each stomp evaluate an importance only for one feature and it is obtained again from the reduction of Gini impurity due to the split on that feature.

Chapter 3

Results

In the rest of this work, we will present the results of the procedures described up to now on the samples that were simulated. In the following table we present a summary of the different design of such populations. We remind that, for each of following set up, half of the populations had a susceptible phenotype, while the other half was considered resistant. The substance that resistance/susceptibility is referred to is and antibiotic of the carbapenem family: the imipenem.

	N. of genomes	Strains per genome	Depth	N. of different designs
Single Specie	1	8	1Gbp	10
Four Specie	4	8	1Gbp	10

Each design was sampled two times, and each time the 8 strains was generated from scratch. In this way we expect to have two populations which are similar, since the strains are created from the same set of genomes, but with actually no genomes in common.

3.1 Single specie populations

In order to start with the simplest possible scenario, we simulated a set of populations containing just one genome. Although sgEvolver diversify it in 8 strains, this scenario is still quite simple and far from the complexity of a real metagenomic sample. The original genomes that was used are 10 different stains of *acinetobacter baumannii*, one in each design. Each design was sampled two times, tor a total of 20 different samples. In the following table we provide the reference to the strains that was used; the AMR phenotype is refered to imipenem.

Strain name	PARTIC ID	AMR phenotype	genome length
Acineto. baumannii BAL_056	470.16198	resistant	4008549
Acineto. baumannii NCSR_106	470.16275	resistant	4123188
Acineto. baumannii A074	470.2428	resistant	4097771
Acineto. baumannii A078	470.2431	resistant	3884708
Acineto. baumannii SIUA14	470.4065	resistant	3939944
Acineto. baumannii NL_6	470.16279	susceptible	3724831
Acineto. baumannii UV_956	470.16281	susceptible	3868975
Acineto. baumannii 91_an	470.16309	susceptible	4003153
Acineto. baumannii 184_n	470.16316	susceptible	3998530
Acineto. baumannii 233_an	470.16326	susceptible	3998495

The read libraries that was generated was aligned to the CARD and WildCARD database in order to obtain the so called resistance spectrum with the procedure described in section 2.3. Those spectra have been filtered with the procedure specified in section 2.4; the following values was used as threshold for the database columns.

- Median: 5 FPKB at least
- Sparsity: present at least in 70% of the samples

In the following plot 3.1(sx) we can have a picture of the effect of the sparsity filter and the median filter on the *resistome* abundances. In this population design, the values chosen for the filter removed 872 ARO identities from the original resistome domain which was of 945 ARO identities. This operation costed the 8% of the sum of all *sample integrals* the samples set. This suggests a reasonable loss of information but a grate dimensional reductions. The loss of *sample integral*, moreover, results to be quite equally distributed among the samples as we can see by the following plot 3.1(dx).

As mentioned, the resistance spectrum data have been standardized and their dimension have been reduced dappling PCA. All the PCA dimensions was taken in to account; since the number of samples, 20, is less then the spectrum dimensions, it will be also the number of our PCA dimensions we consider. In the following bar plot 3.2 is reported the distribution of the variance explained by the PCA components that we obtained. This is an importance measure for the new components. Moreover, we plot the data that we have in the coordinate of the two most important components of PCA with different colors depending on their phenotype. In this way we can have a visualization of how the different samples are distributed in PCA components. In this second picture, we see some clusters of samples, also with uniform colors. Those can't be directly related to the phenotye of the population, but surely hihhlight some similarity between the considered *acinetobacter* strains from which the populations are constructed. We remind that we feed the machine learning algorithms with all the PCA components, so this picture is

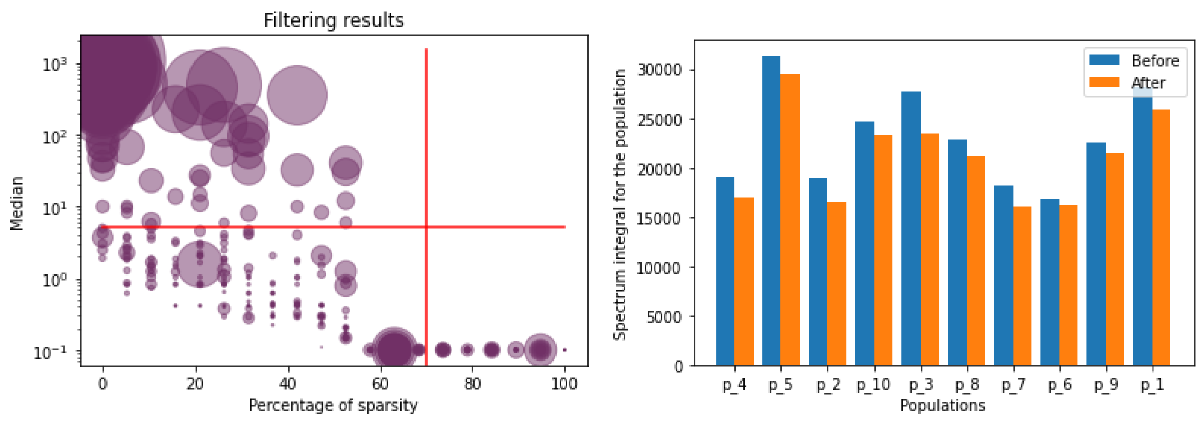


Figure 3.1: In the scatter plot on the left is shown the effects of the sparsity filter and the median one on the *resistance spectra* evaluated from the single specie populations. Each circle represent an ARO identity and the surface of it is proportional to the abundance of that *resistome* in all the populations. The two red lines represents the threshold values for the median and the sparcity. The bar plot on the left represent the spectrum *resistance integral* of each sample before and after the filtering procedure have been applied. The x-axis labels refers to populations name; for graphical reasons, we reported only one of the two samples with the same design.

just a projection of the data we are dealing with. The color of each dot distinguish the phenotype defined for the population.

We trained the classifiers that we mentioned in chapter 2.5 with the result of the PCA analysis; the result of the validation of the classifiers for single species populations are summarized in the following subsections. The correct rate is the amount of samples correctly classified in the validation data-set. Notice that we used a cross validation strategies that perform multiple validations with different splits between validation and training sets. Since, for all the splits, the validation sets never overlap, at the end of the validation process we get one prediction for all of the samples. Correct rate refers then to the ratio between samples correctly classified and the number of all the samples. An other important aspect of the result of cross validation procedure is the distribution of the correct ratio that we obtain from each validation set. Since all the validation splits have the same size, we can say that the correct rate will be the average of such distribution. Anyway, also the shape of this distribution, summarized by its variance, is quite relevant in order to access the stability of our machine learning procedure. Notice that, in the case of Leave one out cross validation, the correct rate will be distributed between two values: 0 and 1. This because in this strategy the validation set is always made by just one sample. We also evaluate the two more parameters which are useful in order to understand what kind of error the algorithm is affected by. False susceptible is

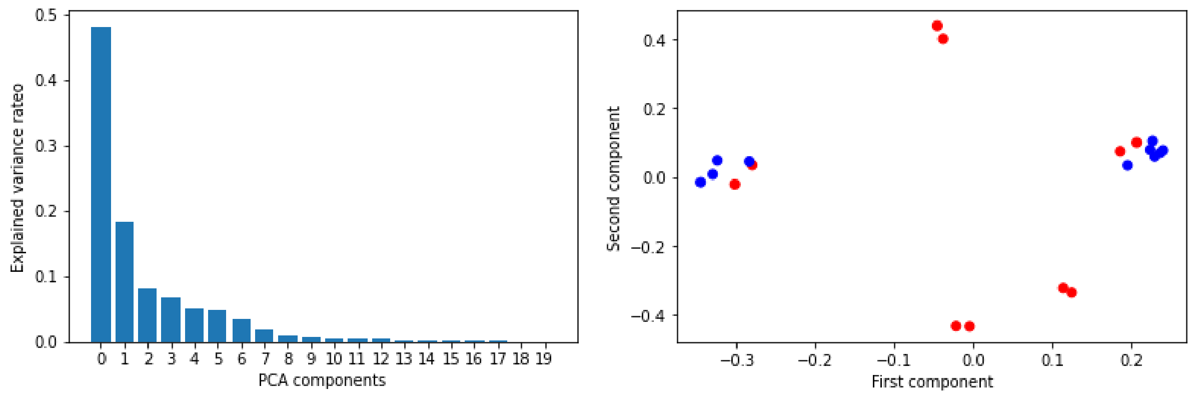


Figure 3.2: In the plot on the right is shown the distribution of the explained variance of our data from various the PCA components. The values on the x-axes labels the new created components. In the plot on the left are represented the resistomes of studied populations in the two first PCA components. The different colors distinguish the population phenotype: RED for ”resistant” and BLUE for ”susceptible”

the number o resistant samples classified as susceptible; false resistant is the number of susceptible samples classified as resistant. Notice that all the cross validation strategies that we exploited assigned just one time each sample to the validation set, so for all of them we have exactly one prediction on their phenotype.

3.1.1 Elastic net logistic regression results for single specie populations

In the following tables are summarized the results of the Logistic regression classifier on the single specie population:

	Five fold cross validation	Leave one out cross val
Average correct rate	0.8	0.7
Correct rate variance	0.06	0.21
False susceptible	4	5
False resistant	0	1

We present also a plot of the correct rate distribution in both validation strategies 3.3; notice that for the leave one out cross validation, the correct rate can be just 1 or 0.

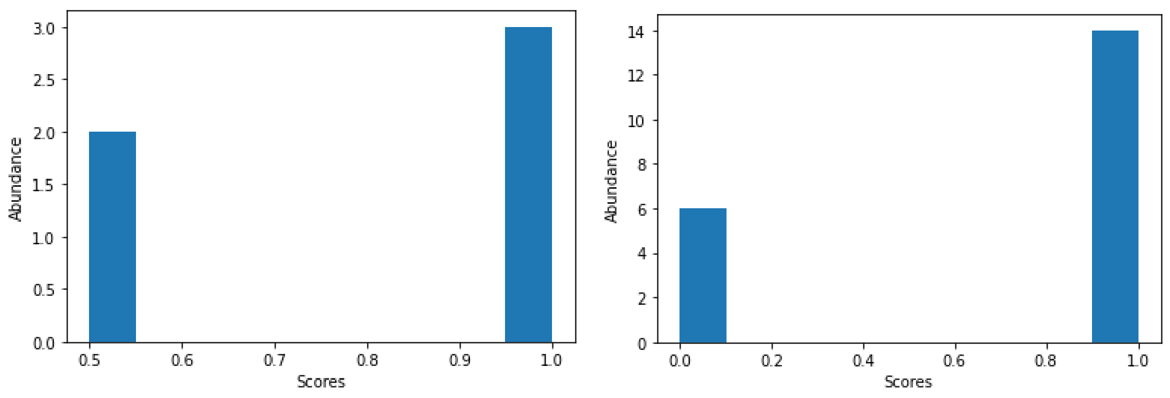


Figure 3.3: In the two bar plots are represented the distribution of the correct rate for five fold cross validation (on the left) and leave one out cross validation (on the right).

3.1.2 Random forest results for single specie populations

In the following tables are summarized the results of the random forest classifier on the single specie population:

	Five fold cross validation	Leave one out cross val
Average correct rate	0.75	0.95
Correct rate variance	0.05	0.05
False susceptible	4	1
False resistant	0	0

We present also a plot of the correct rate distribution in both validation strategies 3.4; notice that for the leave one out cross validation, the correct rate can be just 1 or 0.

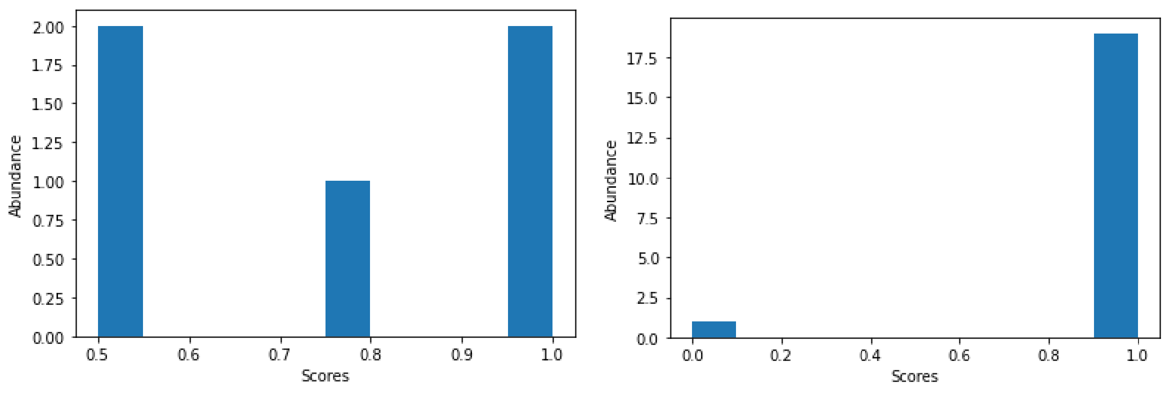


Figure 3.4: In the two bar plots are represented the distribution of the correct rate for five fold cross validation (on the left) and leave one out cross validation (on the right).

3.1.3 AdaBoost results for single specie populations

In the following tables are summarized the results of the AdaBoost classifier on the single specie population:

	Five fold cross validation	Leave one out cross val
Average correct rate	0.8	0.9
Correct rate variance	0.03	0.09
False susceptible	3	1
False resistant	1	1

We present also a plot of the correct rate distribution in both validation strategies 3.5; notice that for the leave one out cross validation, the correct rate can be just 1 or 0.

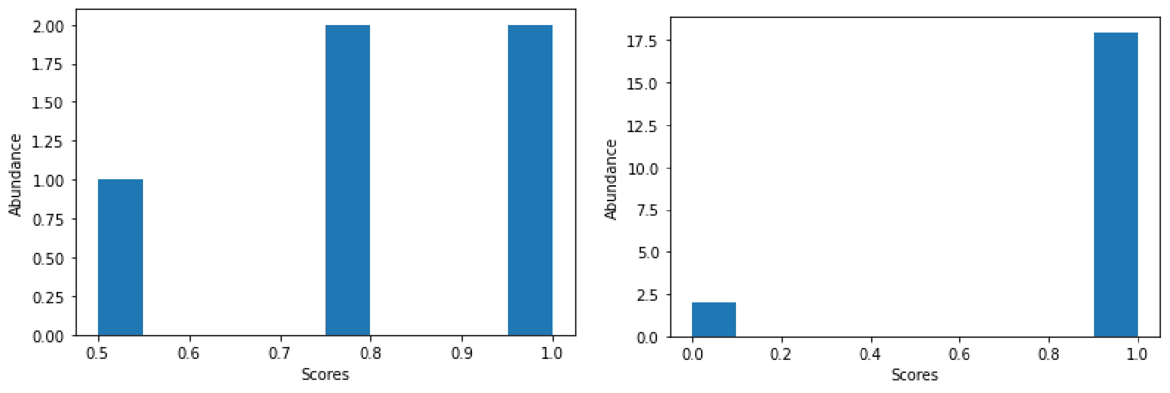


Figure 3.5: In the two bar plots are represented the distribution of the correct rate for five fold cross validation (on the left) and leave one out cross validation (on the right).

3.1.4 Feature importance for single specie populations

In the following 3.6 plots we represent the feature importance that have been obtained from the three different machine learning methods that were exploited, in the two different cross validation strategies. We recall that in this study the features that are taken into account are the PCA components.

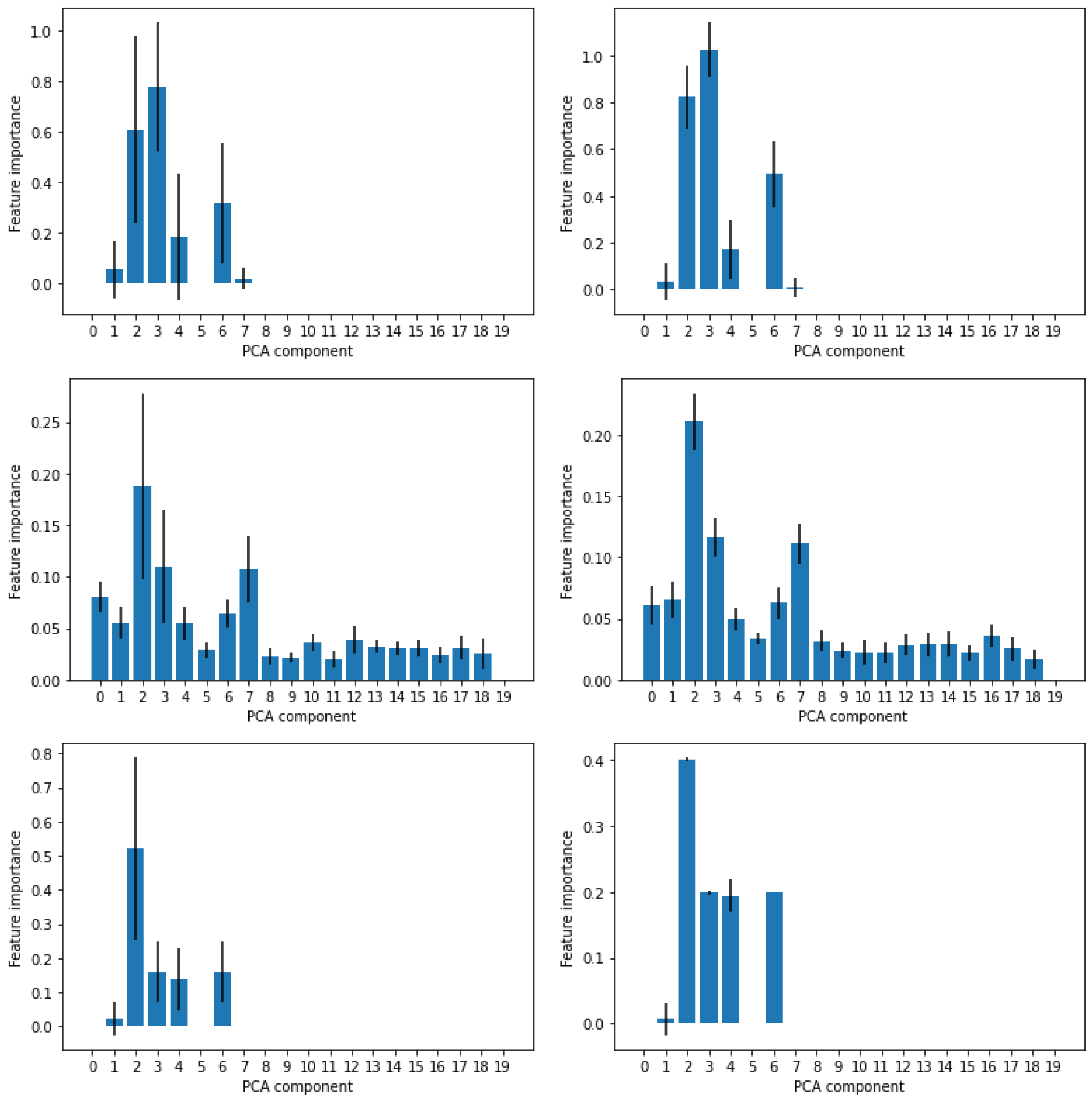


Figure 3.6: From the top to the bottom we have the feature importance evaluated for logistic regression, random forest and AdaBoost. At the right we find leave one out results and at the left five fold cross validation results.

3.2 Four species populations

In order to test our machine learning algorithms in a more complex scenario, samples containing four species was simulated. The genomes that were used as input for Camisim was the same used in the single specie populations. Moreover, we added to each sample three more genomes. Those are the same in each sample and create a sort of background uniform to each sample. All of the three genomes where downloaded from PATRIC and, like the acinetobacter strains, their AMR phenotype relative to the imipenem was noted: all of them are susceptible. In this circumstance, as mentioned in chapter 2.5.1, the population phenotype is determined by the phenotype of the acinetobacter strains: if that is susceptible, all the species in the population are susceptible and so the population is considered susceptible. Otherwise, the acinetobacter will be the only resistant strain in the population, but we will consider the population to be resistant. In the following table we present the references for the genomes used as background. Notice that all the genomes used have been diversified in 8 strains each and the original genome have not been included in the simulation. In the following table, we present the reference to the background genomes that was used.

Specie name	PARTIC ID	AMR phenotype	genome length
Streptococcus pneumoniae	1313.34903	susceptible	1963680
Staphylococcus haemolyticus	1283.799	susceptible	2119443
Yersinia kristensenii	28152.31	susceptible	4448883

The read libraries that was generated was aligned to the CARD and WildCARD database in order to obtain the *resistance spectrum* with the procedure described in section 2.3. Those spectra have been filtered with the procedure specified in section 2.4; the following values was used as threshold.

- Median: 5 FPKB at least
- Sparsity: present at least in 70% of the samples

In the following plot we can have a picture 3.7 of the effect of the sparsity filter and the median one on the *resistance spectrum*. In this population design, the values chosen for the filter removed 768 ARO identities from the original resistome domain which was of 881 ARO identities. This operation costed the 6% of the sum of all *sample integrals* of the samples set, this suggests a reasonable loss of information. This moreover results quite equally distributed as we can see by the following plot.

As mentioned, the resistance spectrum data have been standardized and their dimension have been reduced dappling PCA. All the PCA dimensions was taken in to

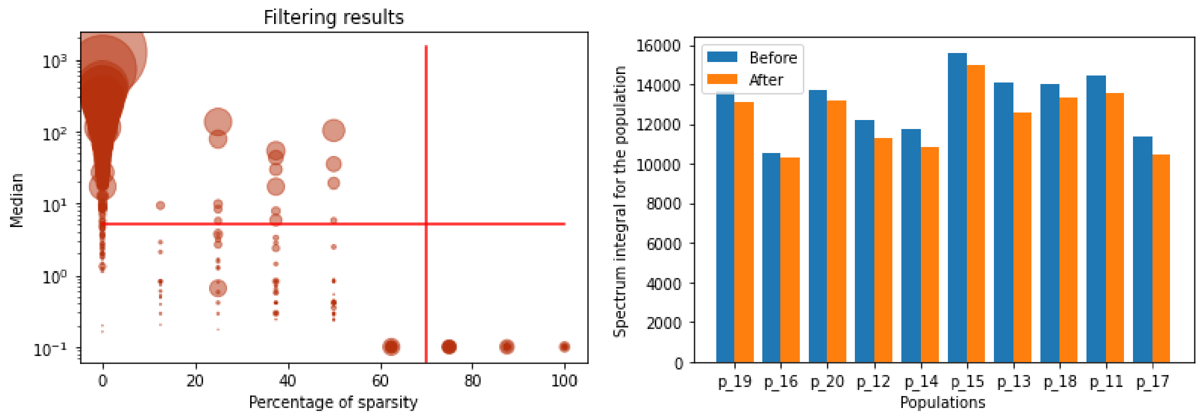


Figure 3.7: In the scatter plot is shown the effects of the sparsity filter and the median one on the resistome evaluated from the four specie populations. Each circle represent an ARO identity and the surface of it is proportional to the abundance of that identity in all the populations. The two red lines represents the threshold values for the median and the sparcity. The bar plot represent the spectrum integral of each sample before and after the filtering procedure have been applied. The x-axis labels refers to populations names.

account; since the number of samples is less then the spectrum dimensions, it will be also the number of our PCA dimensions. In the following bar plot 3.8 is reported the distribution of the variance explained by the PCA components that we obtained. This is an importance measure for the new components.

We plot the data that we have in the coordinate of the two most important components of PCA. In this way we can have a visualization of how the different samples are distributed in PCA components. Te distribution of the points results more sparse then the single population scatter plot, sign of the higher complexity of those samples. We remind that we feed the machine learning algorithms with all the PCA components, so this picture is just a projection of the data we are dealing with. The color of each dot distinguish the phenotype defined for the population.

We trained the classifiers that we mentioned in chapter 2.5 with the result of the PCA analysis; the result of the validation of the classifier are summarized in the following subsections. We recall the considerations made in chapter 3.1 in order to understand those statistics.

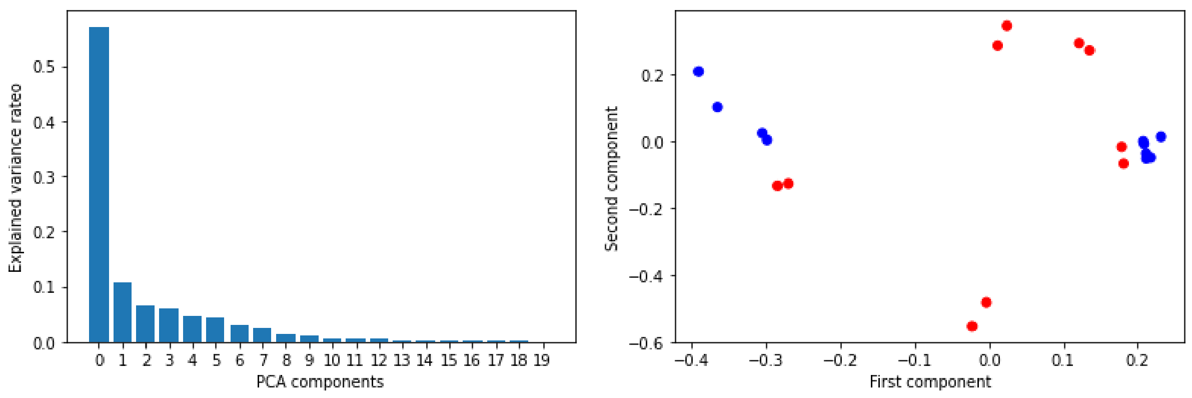


Figure 3.8: In the plot on the right is shown the distribution of the explained variance of our data from various the PCA components. The values on the x-axes labels the new created components. In the plot on the left are represented the *resistomes* of studied populations in the two first PCA components. The different colors distinguish the population phenotype: RED for ”resistant” and BLUE for ”susceptible”

3.2.1 Elastic net logistic regression results for four species populations

In the following tables are summarized the results of the logistic regression classifier on the single specie population:

	Five fold cross validation	Leave one out cross val
Average correct rate	0.85	0.8
Correct rate variance	0.03	0.16
False susceptible rate	1	2
False resistant rate	2	2

We present also a plot of the correct rate distribution in both validation strategies; notice that for the leave one out cross validation, the correct rate can be just 1 or 0 3.9.

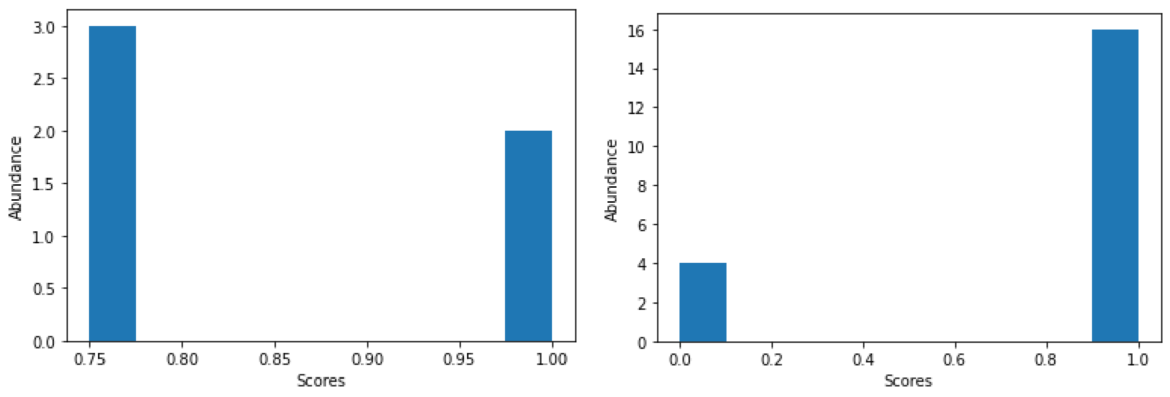


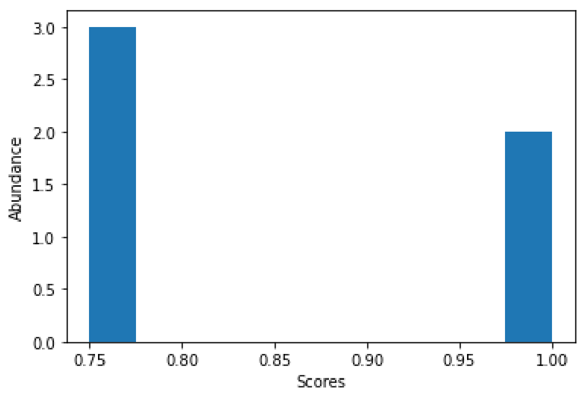
Figure 3.9: In the two bar plots are represented the distribution of the correct rate for five fold cross validation (on the left) and leave one out cross validation (on the right).

3.2.2 Random forest results for four species populations

In the following tables are summarized the results of the random forest classifier on the single specie population:

	Five fold cross validation	Leave one out cross val
Average correct rate	0.85	0.85
Correct rate variance	0.15	0.13
False susceptible rate	1	0
False resistant rate	2	3

We present also a plot of the correct rate distribution in both validation strategies; notice that for the leave one out cross validation, the correct rate can be just 1 or 0 3.10.



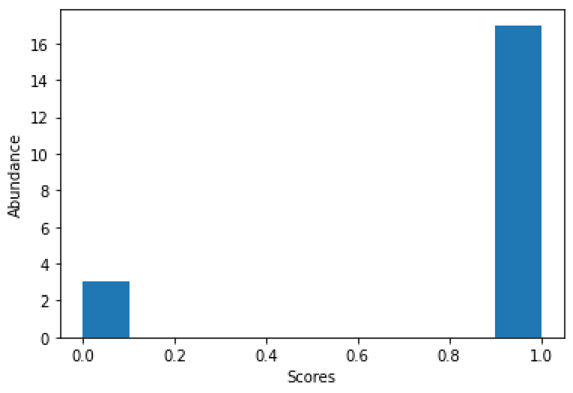


Figure 3.10: In the two bar plots are represented the distribution of the correct rate for five fold cross validation (on the left) and leave one out cross validation (on the right).

3.2.3 AdaBoost results for four species populations

In the following tables are summarized the results of the AdaBoost classifier on the single specie population:

	Five fold cross validation	Leave one out cross val
Average correct rate	0.9	0.9
Correct rate variance	0.02	0.09
False susceptible rate	1	2
False resistant rate	1	0

We present also a plot of the correct rate distribution in both validation strategies; notice that for the leave one out cross validation, the correct rate can be just 1 or 0 3.11.

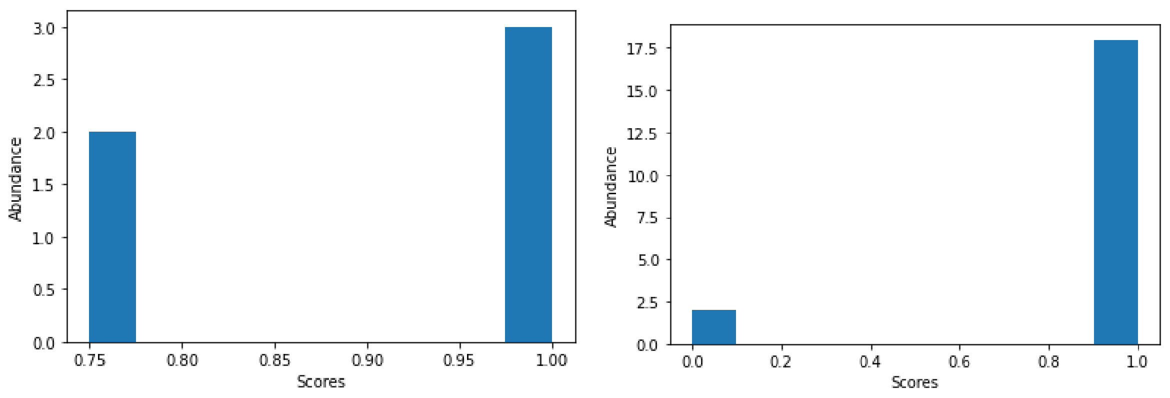


Figure 3.11: In the two bar plots are represented the distribution of the correct rate for five fold cross validation (on the left) and leave one out cross validation (on the right).

3.2.4 Feature importance for four species populations

In the following plots 3.12 we represent the feature importance that have been obtained from the three different machine learning methods that was exploited, in the two different cross validation strategies. We recall that in this study the features that are taken into account are the PCA components.

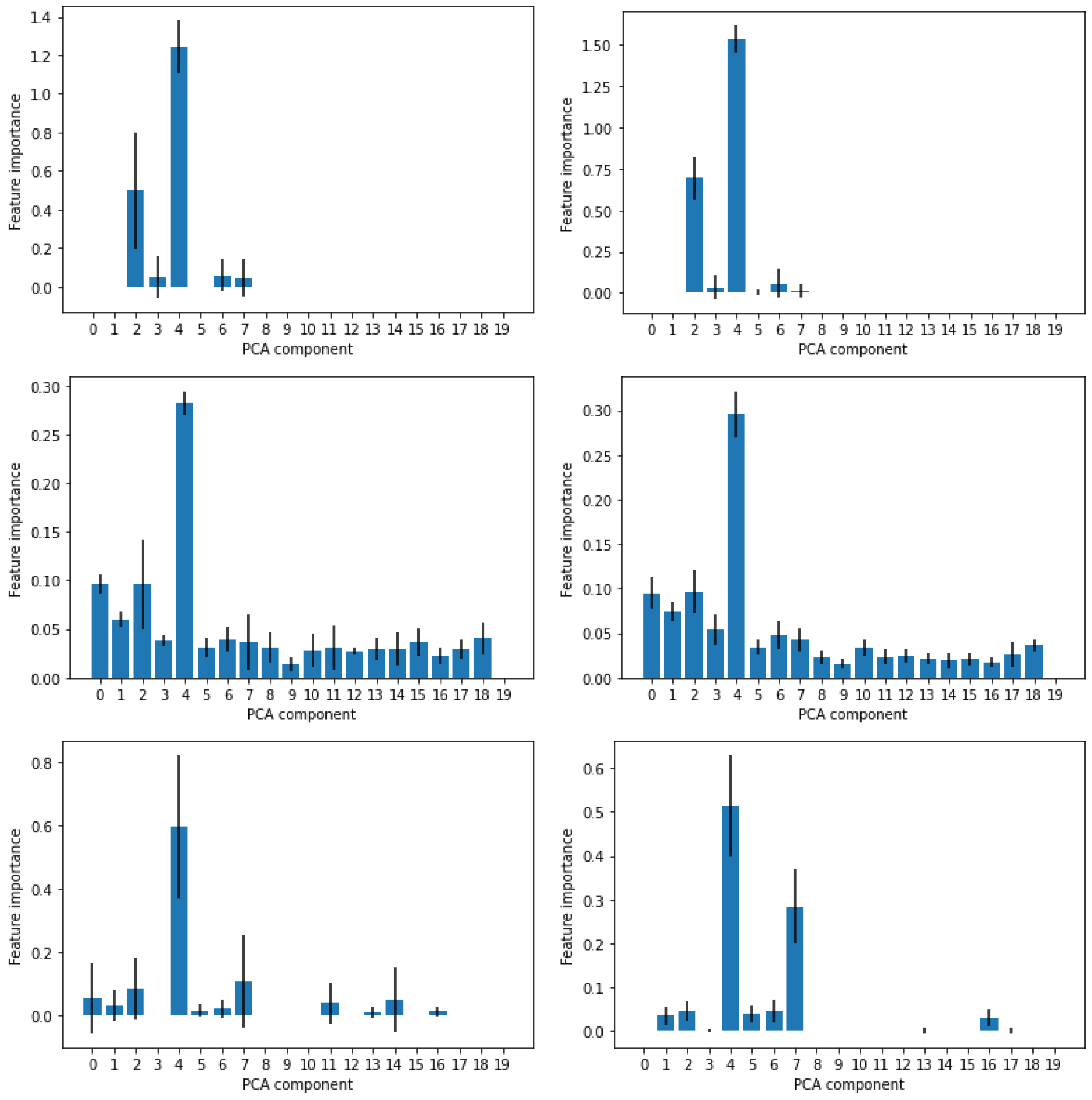


Figure 3.12: From the top to the bottom we have the feature importance evaluated for logistic regression, random forest and AdaBoost. At the right we find leave one out results and at the left five fold cross validation results.

Chapter 4

Conclusions

Since the resistance to antibiotics, it is becoming a serious threat is fundamental to be able to monitor it. The objective of this study was to implement a procedure capable to distinguish resistant bacterial populations from susceptible ones, starting from a metagenomic sample. The resistance we are dealing with is referred to a specific substance: in this study we chose the *Imipenem* antibiotic but in principle any substance could be considered. The only constraint in this sense is availability of a sufficient amount of data in order to train the algorithms. Those data should be metagenomic samples as well, derived from population with known resistance/susceptibility to the substance of interest.

Since we did not have this kind of data, we exploited Camisim in order to generate artificial samples from genomes of bacteria with known resistance/susceptibility. Those genomes where diversified and combined in various way in order to obtain different levels of complexity, but we could not reproduce the diversity of a real bacterial population. Never the less, we consider the simulated scenarios a useful starting point to test our procedure, which well performed on them. Moreover, if the algorithms trained on this artificial data proves to perform well also on real metagenomic data, Camisim could also be used to increase the size of real data training data-sets.

Different population designs with increasing complexity was produced. The more species are included to a population, the more it is comlpex to establish a definition for the resistance/susceptibility for the population. This is a fundamental aspect in order to make possible the classification of them. Two different definitions were tested, corresponding to two different levels of complexity of the samples. In principle any definition could be stated, but surely the performance of the algorithms will strongly depend on it and in some cases those definitions can't even be applied to some scenarios. An example of those limitations is the definition stated for the four species samples: the sample is resistant if there is a resistant specie, otherwise it is susceptible. This definition in principle could be used also for populations containing more species, but there must be a maximum of one resistant species in all populations. This is quite an hard constraint and surely a more comprehensive definition has to be found in order to analyze more complex samples. A solution to this limitation could be to perform a taxonomic classification of the reads in the samples. In this way, the subsets of the original metagenomic sample could be analyzed independently, assuming to be single specie populations and the resistant/susceptible definition would be straightforward. The taxonomic classification is quite a standard analysis [23] and usually is capable of classifying the reads at the specie level, but in some case can also distinguish different strains. In this way we would have very simple populations to study and the phenotype definition problem would become trivial.

The implemented procedure exploits an innovative feature extraction strategy in order to provide the data to the machine learning algorithms. The reads where aligned to two databases containing genes associated to various resistance mechanisms (*resistomes*) and the data provided to the machine learning algorithms was the estimated abundance of such genes. We named this multidimensional quantity *resistance spectrum.* This procedure is very specific for antibiotic resistance study and requires to have a reference database containing the resistance genes (*resistomes*): we used CARD and WildCARD databases.

Both for the single species and the four species design, we filtered the *resistance spectra* of 20 samples, and reduced their dimension with PCA. Three machine learning algorithms were trained with those data and their performance was evaluated with cross validation. Result are detailed chapter 3 and here we provide a summary: in the following tables is reported the average score of the classifiers on the different validation sets.

- Leave one out cross validation results

Classifier	Single specie score	Four species score
Logistic regression	0.72 ± 0.45	0.8 ± 0.4
Random forest	0.95 ± 0.22	0.85 ± 0.4
AdaBoost	0.9 ± 0.3	0.9 ± 0.3

- Five fold cross validation results

Classifier	Single specie score	Four species score
Logistic regression	0.8 ± 0.24	0.85 ± 0.17
Random forest	0.75 ± 0.22	0.85 ± 0.38
AdaBoost	0.8 ± 0.17	0.9 ± 0.14

All three tested methods seem to perform well in the classification of the samples. The scores with different validation techniques and on different design of population do not present a clear ranking of the classifiers, but we can say that AdaBoost has an overall better power in distinguishing susceptible from resistant samples. Its performance are also quite stable as we can see by the low standard deviation values. This means that the performance of the classifier do not depend in a strong way on which samples we use for the training and which ones we use for the validation and is a good indicator of the stability of the algorithm. In order to see more clearly which machine learning algorithms performs better, would probably be required a larger collection of samples. The result of this ranking is likely to be dependent also no the parameters setting of the classifiers. Further analysis will be carried on in this sense in order to identify the best procedure to perform this classification. Anyway, we should say that the performance already obtained with such a small sample size are quite satisfactory.

An interesting comparison that will be surely performed is instead between the performances achieved with this feature extraction procedure and the performance of more canonical approaches. Usually the read libraries are decomposed in k-mers and then the count of those k-mers is used a input for the machine learning algorithms [24]. This procedure is more generic and is useful also for other classification procedures that don not regard AMR. Anyway, specifically for AMR study, the feature extraction procedure should take into account the knowledge about resistance genes that is collected in databases like CARD. For this reason we expect that the the resistance spectrum should be the right way to extract the features from read libraries when the objective is accessing AMR phenotype.

The two population design that was simulated have a great difference in terms of complexity, but the performance of the classifiers don not seems to be affected by this. All three algorithms are capable of classifying correctly the population, both if they are simulated starting from a single genome or four genomes. We remind that three of the four genomes, in the four species design, are kept constant in all populations (even if we sample the reads from the strain produced by sgEvolver which are different from time to time) creating a sort of background noise. The classifiers that we tested were able to overcome this difficulty identifying the variation of the acinetobacter genome and also the phenotype of those genomes. This let us being quite optimistic for the future developments of this study: the composition of the feature extraction procedure and the machine learning algorithms don not seem to be affected by an increase of sample complexity. An important part of the future development of this study will verify this statement and test the procedures also on more diverse sample design in order to approach the complexity of real metagenomic samples.

Finally, the feature importance profiles evaluated from the machine learning procedures highlight how the different algorithms, although differently implemented, agree on the importance of a certain feature (second component for single species population and fourth component of the PCA). The other features results still relevant: the random forest classifier takes advantage of almost all of them while the logistic regression and the AdaBoost classifier exploits just a subset. Understanding why the second or fourth

PCA component is so important in the classification process goes beyond the objectives of this study, but it will be the subject of further developments. It is likely that a more clarifying picture could be provided without performing the PCA and this will be also a point that will be deepen in future works.

Chapter 5

Ringraziamenti

Questo elaborato è il completamento di un' esperienza durata sei anni che le persone chiamano università. In questi sei anni sono successe svariate cose oltre all'università, ma questa ha dettato il tempo di tutte le altre, costringendomi a non essere troppo sfaticato o troppo selvaggio. Finita l'università, quindi, mi aspetto che qualcosa cambi, che qualcosa di nuovo cominci e che qualcosa di vecchio finisca. In questa situazione è bello lasciarsi andare a un po' di nostalgia per il tempo passato e ricordare le persone che lo hanno reso memorabile. Concedetemi quindi qualche pagina per menzionare chi di dovere e concedetemi di farlo in italiano, che l' inglese è adatto al massimo a dei batteri.

Innanzitutto ringrazio Claudia, Ettore, Gastone, Daniel, Nicolas, Alessandro... che mi hanno introdotto e appassionato alle tematiche della mia tesi e dei miei studi futuri. Con voi ho scoperto una dimensione sociale della scienza, di scambio e collaborazione, che ha rimpiazzato lo studio solitario e alienante a cui ero abituato. Sono molto contento di poter' continuare a lavorare con voi ancora per un po'.

Oppostamente, ringrazio anche tutti i colleghi degli improbabili lavori che ho fatto: vendemmiatore, cameriere, bagnino, venditore allo stadio, etc... Siete troppi da nominare e alcuni di voi mi stanno pure sull'anima, ma esservi colleghi mi ha dato molto più del misero stipendio. Mi avete insegnato la varietà dell'uomo e ad andar d'accordo quasi con tutti quando c'è da lavorare. Questo genere di occupazioni non penso avrà più posto nella mia vita, almeno per un po'.

Ringrazio poi i miei amici, che in modo diverso mi hanno abbellito l'esistenza, in questi sei anni e prima. Questi, invece, vorrei nominarli tutti, ma mi conterrò e spero che gli esclusi non se ne abbiano a male.

I primi che ringrazio sono Francesco, Francesco, Federico, Lara, Francesca, Dario, Andrea... In un film [25] che vi consiglio si dice che con certe persone si raccontano solo le cose belle. Voi siete decisamente quelle persone; ci siete da sempre e ci sarete il più a lungo possibile.

Alessandro, Domenico, Mattia, Luca, Ciro, Chiara e Cecilia... con voi invece ho condiviso anche dispiaceri e bile. Eppure ci siamo divertiti parecchio e sono sicuro che continueremo a farlo. Faremo casino e ci rilasseremo a piacimento, limitandoci solo quando siamo noi a volerlo e non quando sarebbe conveniente.

Ringrazio Chiara, una strana creatura che da un po' mi segue, oppure io seguo lei. Non ho ben' capito, ma la sua vena artistica è di ispirazione e spero impari a godertela.

Ringrazio Elena, Michela, Davide, Alessandra, Fabrizio, Chiara, Gabriele, Giordano, Ilaria, Lorenzo, Lorenzo, Lorenzo, Pietro, Matilde, Matteo, Nicøla, Pietropaolo, Davide, Mattia... Mi avete regalato una città e poi l'Italia intera. Siete sparsi un po' in giro, ma farò il possibile per continuare a trovarvi e continuare a vivere avventure mirabolanti.

Ringrazio anche le ragazze che, a modo mio, ho amato. Non ci provo neanche a spiegare, ne a buttare lì dei nomi: questo ringraziamento è già abbastanza controverso. Eppure sento che siete state e contninuerete ad essere un parte importante della mia vita. Il ricordo del tempo passato con voi è una grande ricchezza e l'aspettativa di qualsiasi futuro è un sogno.

Ringrazio infine la mia famiglia, un' entità che ha trè nomi che non pronuncio quasi mai: Franca, Piero e Antonio. Siete i miei idoli ed i miei benefattori. Voi più di tutti mi avete preparato per questo viaggio e avete soffiato sulle mie vele bizzarre, dandomi piena fiducia sulla rotta. Sento che sta per arrivare un' onda che mi spingerà parecchio al largo, ma coglierò ogni occasione per tornare a raccontarvi quello che ho visto e ascoltare cosa avete da dire. Ne parleremo a tavola mangiando qullo che abbiamo cucinato e bevendo buon vino.

Bibliography

[1] Antimicrobial Resistance Collaborators. Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis. *Lancet*, 399(10325):629–655, February 2022.

[2] O. T. Avery, C. M. Macleod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types : Induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type iii. *The Journal of experimental medicine*, 79(2):137–158, Feb 1944. 19871359[pmid].

[3] J. D. WATSON and F. H. C. CRICK. Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, Apr 1953.

[4] F Sanger, E O Thompson, and R Kitai. The amide groups of insulin. *Biochem. J.*, 59(3):509–518, March 1955.

[5] F Sanger, G M Air, B G Barrell, N L Brown, A R Coulson, C A Fiddes, C A Hutchison, P M Slocombe, and M Smith. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, 265(5596):687–695, February 1977.

[6] R D Fleischmann, M D Adams, O White, R A Clayton, E F Kirkness, A R Kerlavage, C J Bult, J F Tomb, B A Dougherty, and J M Merrick. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 269(5223):496–512, July 1995.

[7] Centers for Disease Control, Prevention, et al. *Antibiotic resistance threats in the United States, 2019*. US Department of Health and Human Services, Centres for Disease Control and . . . , 2019.

[8] Adriano Zaghi and Ettore Rocchi. https://github.com/adrianozaghi/amr.

[9] James J Davis, Alice R Wattam, Ramy K Aziz, Thomas Brettin, Ralph Butler, Rory M Butler, Philippe Chlenski, Neal Conrad, Allan Dickerman, Emily M Dietrich, Joseph L Gabbard, Svetlana Gerdes, Andrew Guard, Ronald W Kenyon,

Dustin Machi, Chunhong Mao, Dan Murphy-Olson, Marcus Nguyen, Eric K Nordberg, Gary J Olsen, Robert D Olson, Jamie C Overbeek, Ross Overbeek, Bruce Parrello, Gordon D Pusch, Maulik Shukla, Chris Thomas, Margo VanOeffelen, Veronika Vonstein, Andrew S Warren, Fangfang Xia, Dawen Xie, Hyunseung Yoo, and Rick Stevens. The PATRIC bioinformatics resource center: expanding data and analysis capabilities. *Nucleic Acids Res.*, 48(D1):D606–D612, January 2020.

[10] Luísa C.S. Antunes, Paolo Visca, and Kevin J. Towner. Acinetobacter baumannii: evolution of a global pathogen. *Pathogens and Disease*, 71(3):292–301, 08 2014.

[11] Ioannis Kyriakidis, Eleni Vasileiou, Zoi Dorothea Pana, and Athanasios Tragiannidis. Acinetobacter baumannii antibiotic resistance mechanisms. *Pathogens*, 10(3):373, March 2021.

[12] Adrian Fritz, Peter Hofmann, Stephan Majda, Eik Dahms, Johannes Dröge, Jessika Fiedler, Till R Lesker, Peter Belmann, Matthew Z DeMaere, Aaron E Darling, Alexander Sczyrba, Andreas Bremges, and Alice C McHardy. CAMISIM: simulating metagenomes and microbial communities. *Microbiome*, 7(1):17, February 2019.

[13] A. Darling, M. Craven, B. Mau, and N.T. Perna. Multiple alignment of rearranged genomes. In *Proceedings. 2004 IEEE Computational Systems Bioinformatics Conference, 2004. CSB 2004.*, pages 738–739, 2004.

[14] Robert MacArthur. On the relative abundance of species. *The American Naturalist*, 94(874):25–36, 1960.

[15] Weichun Huang, Leping Li, Jason R. Myers, and Gabor T. Marth. ART: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 12 2011.

[16] Fastqc, Jun 2015.

[17] Fastqc manual, Oct 2018.

[18] Brian P Alcock, Amogelang R Raphenya, Tammy T Y Lau, Kara K Tsang, Mégane Bouchard, Arman Edalatmand, William Huynh, Anna-Lisa V Nguyen, Annie A Cheng, Sihan Liu, Sally Y Min, Anatoly Miroshnichenko, Hiu-Ki Tran, Rafik E Werfalli, Jalees A Nasir, Martins Oloni, David J Speicher, Alexandra Florescu, Bhavya Singh, Mateusz Faltyn, Anastasia Hernandez-Koutoucheva, Arjun N Sharma, Emily Bordeleau, Andrew C Pawlowski, Haley L Zubyk, Damion Dooley, Emma Griffiths, Finlay Maguire, Geoff L Winsor, Robert G Beiko, Fiona S L Brinkman, William W L Hsiao, Gary V Domselaar, and Andrew G McArthur. CARD 2020: antibiotic resistome surveillance with the comprehensive antibiotic resistance database. *Nucleic Acids Res.*, 48(D1):D517–D525, January 2020.

[19] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature Methods*, 9(4):357–359, April 2012.

[20] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[21] Hastie Trevor, Tibshirani Robert, and Friedman Jerome. The elements of statistical learning: data mining, inference, and prediction, 2009.

[22] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[23] Derrick E. Wood, Jennifer Lu, and Ben Langmead. Improved metagenomic analysis with kraken 2. *Genome Biology*, 20(1):257, Nov 2019.

[24] Sanjat Kanjilal Melis N Anahtar, Jason H Yang. Applications of machine learning to the problem of antimicrobial resistance: an emerging model for translational research. *Journal of clinical microbiology vol. 59,7 (2021)*, 59(7):119–139, 2021.

[25] Youth - la giovinezza, 2015.